# Intelligent System Synthesis for Dynamic Locomotion Behavior in Multi-legged Robots

*Thesis submitted for*
*acquiring the degree*

*of*

Master of Engineering

*by*


Azhar Aulia Saputra

Under the guidance of
Professor Naoyuki Kubota

*Department of Intelligent Mechanical System*
*Graduate School of System Design*
*Tokyo Metropolitan University*
*Tokyo, Japan*


February, $6^{th}$ 2018

# Abstract

Robot technology has been implemented in many fields of our life, such as entertainment, security, rescue, rehabilitation, social life, the military, and etc. Multi-legged robot always exist in many fields, therefore it is important to be developed. Motion capabilities of the robot will be a main focus to be developed. Current development or conventional model of motion capabilities have several issues in saturation of development. There are some limitation in dynamic factors such as, locomotion generator, flexibility of motion planning, and smoothness of movement. Therefore, in this research, natural based computation are implemented as the basic model. There are three subsystems to be developed and integrated, (1) locomotion behavior model, (2) stability behavior model, and (3) motion planning model.

Since individual people has different walking behavior in each walking direction and walking speed, locomotion behavior learning model of omni-directional bio-inspired locomotion which is generating different walking behavior in different walking provision are required to be developed. Step length in sagital and coronal direction, and degree of turning are considered parameters in walking provision. In proposed omni-directional walking model, interconnection structures composed by 16 neurons where 1 leg is represented by 4 joints and 1 joint is represented by 2 motor neurons. In order to acquire walking behavior in certain walking provision, the interconnection structure is optimized by multi-objectives evolutionary algorithm. For acquiring the diversity of references, several optimized interconnection structures are generated in optimization processes in different walking provisions. Learning models are proposed for solving non-linearity of relationship between walking input and walking output representing the synaptic weight of interconnection structure, where one learning model representing one walking parameter. Furthermore, by using optimized model, walking behavior can be generated with unscaled walking provision. Smooth walking transition with low error of desired walking provision was proved based on several numerical experiments in physical computer simulation.

In stability behavior model, neuro-based push recovery controller is applied in multi-legged robot in order to keep the stability with minimum energy required. There are three motion patterns in individual people behavior when it gets external perturbation, those are

ankle behavior, hip behavior, and step behavior. We propose a new model of Modular Recurrent Neural Network (MRNN) for performing online learning system in each motion behavior. MRNN consists of several recurrent neural networks (RNNs) working alternatively depending on the condition. MRNN performs online learning process of each motion behavior controller independently. The aim of push recovery controller is to manage the motion behavior controller by minimizing the energy required for responding to the external perturbation. This controller selects the appropriate motion behavior and adjusts the gain that represent the influence of the motion behavior to certain push disturbance based on behavior graphs which is generated by adaptive regression spline. We applied the proposed controller to the humanoid robot that has small footprint in open dynamics engine. Experimental result shows the effectiveness of the push controller stabilizing the external perturbation with minimum energy required.

Proposed motion planning model presents a natural mechanism of the human brain for generating a dynamic path planning in 3-D rough terrain. The proposed model not only emphasizes the inner state process of the neuron but also the development process of the neurons in the brain. There are two information transmission processes in this proposed model, the forward transmission activity for constructing the neuron connections to find the possible way and the synaptic pruning activity with backward neuron transmission for finding the best pathway from current position to target position and reducing inefficient neuron with its synaptic connections. In order to respond and avoid the unpredictable obstacle, dynamic path planning is also considered in this proposed model. An integrated system for applying the proposed model in the actual experiments is also presented. In order to confirm the effectiveness of the proposed model, we applied the integrated system in the pathway of a four-legged robot on rough terrain in computer simulation. For analyzing and proving the flexibility of proposed model, unpredictable collision is also performed in those experiments. The model can find the best pathway and facilitate the safe movement of the robot. When the robot found an unpredictable collision, the path planner dynamically changed the pathway. The proposed path planning model is capable to be applied in further advance implementation.

In order to implement the motion capabilities in real cases, all subsystem should be integrated into one interconnected motion capabilities model. We applied small quadruped robot equipped with IMU, touch sensor, and dual ultrasonic sensor for performing motion planning in real terrain from starting point to goal point. Before implemented, topological map is generated by Kinect camera. In this implementation, all subsystem were analyzed and performed well and the robot able to stop in the goal point. These implementation proved the effectiveness of the system integration, the motion planning model is able to generate safe path planning, the locomotion model is able to generate flexible movement depending on the

walking provision from motion planing model, and the stability model can stabilize the robot on rough terrain. Generally, the proposed model can be expected to bring a great contribution to the motion capabilities development and can be used as alternative model for acquiring the dynamism and efficient model in the future instead of conventional model usage.

In the future, the proposed model can be applied into any legged robot as navigation, supporter, or rescue robot in unstable environmental condition. In addition, we will realize a cognitive locomotion that generates multiple gaits depending on the 3 aspects, embodiment, locomotion generator, and cognition model. A dynamic neuro-locomotion integrated with internal and external sensory information for correlating with the environmental condition will be designed.

平成29年度　修士論文

# Intelligent System Synthesis for Dynamic Locomotion Behavior in Multi-legged Robots

首都大学東京大学院
システムデザイン研究科　博士前期課程
知能機械システム学域

学修番号　16889503

Azhar Aulia Saputra

指導教員　久保田　直行　教授

平成30年2月

# 概要

ロボット技術は、エンターテイメント、セキュリティ、救助、リハビリ、社会生活、軍事などの様々な生活分野に実現さている。多脚ロボットは常に多くの分野に存在するため開発することが重要である。ロボットの運動能力が開発の主要となっている。現状の開発されている動作能力は，飽和状態にある。いくつかの動的な要因により、歩行生成器、動作計画の柔軟性、および動作の滑らかさ等に制限がある。そこで、本研究では、基本的なモデルとして自然計算に基づく方法論を実装する。また、本研究では、歩行動作モデル、安定動作モデル、や運動計画モデルからなる３つのサブシステムを開発し統合する。

人間は歩行方向と速度に応じて歩行動作が異なるため、異なる歩行軸では異なる歩行動作を生成するという全方位生物的な運動の歩行動作学習モデルが開発には要求される．球欠および制御方向のステップ長や旋回の度合いは，歩行軸のパラメータとして考慮される。提案した全方位歩行モデルでは，1肢につき16個のニューロンによって構成される相互接続構造を4つの関節によって表現する．また、1つの関節は，2個のモータニューロンによって表現する．一定の歩行軸での歩行動作を獲得するために，本研究では，多目的進化アルゴリズムによって最適化を行う。提案手法では、参照点の多様性を獲得するために，異なる歩行軸においていくつかの最適な相互接続構造が生成される。相互接続構造のシナプス重みを表現している歩行入力と出力間の非線形な関係を解くための学習モデルを構築する．本手法では，1つの学習モデルが1つの歩行パラメータで表現され、最適化されたモデルを用いることにより，歩行動作は，スケーリングされていない歩行軸を生成することが可能となる。物理演算シミュレーションを用いた実験により，誤差の少ない歩行軸の滑らかな歩行遷移を本実験では示している。

安定動作モデルでは、必要最小限のエネルギーで安定性を維持するため多足歩行ロボットにニューロベースプッシュリカバリ制御器を適用した。外力をを受けたとき，人間の行動には足首の動作・股関節の動作・踏み動作の3つの動作パターンが存在する。本研究では，各運動動作におけるオンライン学習システムを実現するために、モジュラーリカレントニューラルネットワーク（MRNN）を用いた新たな学習モデルを提案する。MRNNは状況に応じて選択される複数のリカレン

トニューラルネットワーク（RNN）によって構成される。MRNNは各運動動作コントローラのオンライン学習プロセスを独立して実行する。プッシュリカバリ制御器の目的は、外乱に応じてエネルギー最小化を行うことによって運動動作制御器を管理することである。この制御器は適切な運動動作を選択し，適応回帰スプラインにより生成された動作グラフに基づき押し動作に対して最も影響を及ぼす運動動作のゲインの調整を行う。提案した制御器をOpen Dynamics Engine（ODE）上で小さな足の長さを持つヒューマノイドロボットに適用し，必要最小限のエネルギーで外力に対して安定させるプッシュリカバリ制御器の有効性を示している。

3次元の不整地における動的な経路計画を生成するために，人間の自然な脳機能に基づいた動作計画手法を提案する。本モデルは、ニューロンの内部状態過程だけでなく、脳内のニューロンの発達過程も重視している。本モデルは二つのアルゴリズムに構成される。1つは、通過可能な道を見つけるために構築される接続的なニューロン活動である順方向伝達活動であり，もう1つは、現在位置から最適経路を見つけるために、シナプス結合を用いて非効率的なニューロンを減少させる逆方向にニューロン伝達を行うシナプスプルーニング活動である。また，予測不可能な衝突を回避するために，動的な経路計画も実行される。さらに、実環境において提案されたモデルを実現するための統合システムも提示される。提案モデルの有効性を検証するために，コンピュータシミュレーション上で、不整地環境の4足歩行ロボットに関するシミュレーション環境を実装した。これらの実験では，予測不能な衝突に関する実験も行った。本モデルは、最適経路を見つけ出しロボットの安全な移動を実現できた。さらに、ロボットが予測できない衝突を検出した場合，経路計画アルゴリズムが経路を動的に変更可能であることを示している。これらのことから、提案された経路計画モデルはさらなる先進的な展開が実現可能であると考えられる。

実環境における運動能力を実装するためには、すべてのサブシステムを1つの運動能力モデルに統合する必要がある。そこで本研究では、IMU、タッチセンサ、2つの超音波センサを搭載した小型の4足歩行ロボットを用いた実環境において出発地点から目的地点までの運動計画を行った。本実装では、3次元距離計測センサであるKinecを用い3次元空間の位相構造を生成する。また、本実装では、すべてのサブシステムが分析され、ロボットは目的地点で停止することができた。さらに、安全な経路計画を生成することができたことからシステム統合の有効性が確認できた。また、歩行モデルにより歩行軸に応じた柔軟な動きが生成されることで、この安定性モデルは不整地環境でもロボットの歩行を安定させることができた。これらのことから、本提案モデルは運動能力への多大な貢献が期待され、ダイナミクスを獲得するための代替モデルとして使用することができ，現在

よく使用されているモデルに代わる効率的なモデルとなることが考えられる．

今後の課題としては，不安定な環境下におけるナビゲーション・支援・レスキューロボットといった任意の肢の数を持つ多足歩行ロボットへの本提案モデルの適用があげられる．さらに，身体性，歩行生成，認知モデルの3つの観点から複数の歩容を生成する認知的歩行を実現することを考えている．環境と相互作用するためのモデルとして、内界センサと外界センサ情報を統合した動的ニューロ歩行を実現する予定である．

# Acknowledgement

I would like to give my humble thanks to Allah, because of His blessing and benevolence I could live until now and reach .

I would like also to present this thesis to my wife, Ashri Ratnasari and our parents. Their struggle to raise and support me, have taken me until this point. And also my little brother, many thanks for his supports and spirit during my life in Japan.

I would like to extend my special thanks to my advisor, Prof. Naoyuki Kubota for his kind and strong supports, inspiring advises, and wonderful guidance. I could learn and experience a lot of things from him in the past three years and half.

I would like to thank to Dr. Janos Botzheim, his advices and supports gave me the opportunity to learn about metodology and paper writing. For Mr. Indra Adji Sulistijono, I would like to thank for his advices and recomendations for studying in Japan

I would also like to thank to Mr. Achmad Subhan Khalilullah, Mr. Ali Husein Alasiry, and All of my supervisors in Politeknik Elektronika Negeri Surabaya for his advices and supervision during my undergraduate so that I could receive great achievements as consideration study in Japan.

Thanks to all the members of the laboratory for their support and advice. Ms. M. Shimada and Ms. T. Suzuki kindly supported my daily life in the laboratory. Dr. Yuichiro Toda, Dr. Noel Tay, Dr. Jinseok Woo, Dr. Takahiro Takeda and Dr. Takenori Obo for their advices and support during conducting researches and projects. Mr. Kaif Jaden Chin, Mr Iwasa for being good friend, partner, with nice advices. Mr Sun Siqi, Mr Shao Shuai, and Ms. Chiaki Kasuya for helping me during lecture, sharing information, report writing, and daily support. Mr. Shuo, Ms. Noriko, and Mr. Quan for their support. Mr. Shimazaki, Mr. Yoshida, Mr. Matsuo, and Mr. Kusaka for their friendly support. Mr. Kaku, Ms. Yuri, and Ms. Ono for helping me in doing projects. Mr. Miki, Mr. Mabuchi, Mr. Miyake, Mr. Zheng Wei, Mr. Tamatsu, Mr. Sato, and Mr. Joji Sato for their daily support in the laboratory. Mr. Kato, Mr. Sasuga, Mr. Tonomura, Mr. Kanei, Mr. Arakawa, Mr. Arai, Mr. Nakamura, Mr. Shimizu, Mr. Tanaka, Mr. seaside, and Mr. Wu Zepei for having good conversation. Lastly, Also many thanks to Mr. Yusuf Bakhtiar for his helps in Japan life, private life, Japanese translation.

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

## 1.1 Background

### 1.1.1 Social background

Nowadays, robots are no longer utilized as amere technological supplementary tools for labor-intensive or hazardous tasks such as, factory automation, military operation, and even space or seabed exploration). Now, robot has become a part of our daily lives. Robot technology has been implemented in many fields of our life, such as entertainment, security, rescue, rehabilitation, social life, industry, and the military. Most researchers build robots for particular purposes. Some researchers use tank model robots for disaster problems and navigation in dangerous areas [228], while other researchers build a robot partner to support elderly people [233]. Furthermore, some researchers use humanoid robots for dangerous areas and rescuing humans [207]. Honda produced the humanoid robot "ASIMO" that can serve people in their social life. DARPA developed a humanoid robot for military service. Nevertheless, the humanoid biped robot is a suitable robot in many fields: it can be applied for social life [167], rescue [228], [207], military purposes, or entertainment (Soccer Robot, Dancing Robot) [176], [173]. Therefore, it is important to improve the development of legged robots. Although the cognitive capabilities of humanoid robots are important, but their motion capabilities are also important to support its activity. Therefore, in this thesis, we developed motion capabilities of the robot. In our motion capabilities model, there are three subsystems should be developed, which are, (1) locomotion behavior model, (2) stability behavior model, and (3) motion planning model.

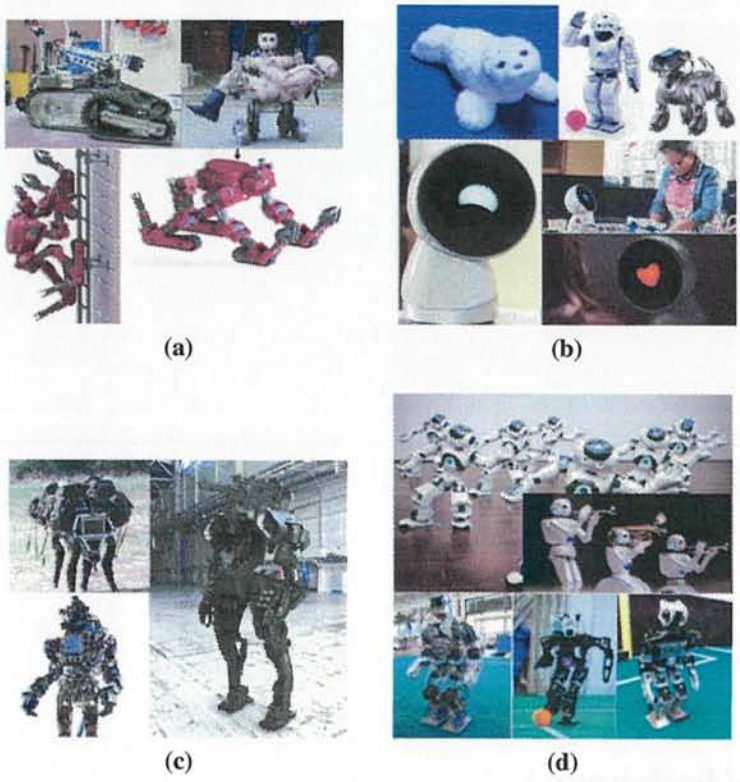



**Figure 1.1:** *(a) Rescue robots (b) Social robots (c) Millitary robots (d) Entertainment robots*

### 1.1.2 Technical background

In motion capabilities model, most of the researcher using conventional model to develop trajectory generation that used zero moment point and the inverted pendulum approach [176], [173], [234]. Yoshida et al. implemented the double inverted pendulum for the balancing system in their humanoid robot [234]. Most of them develop the certain trajectory generator for certain motion behavior [96, 28, 132, 193, 32, 121]. Kim et al. also used a conventional approach for development of the HUBO robot: they applied a mathematical method to develop the trajectory generator in the robot [96]. They developed difference mathematical model for developing 3 difference walking behavior (Sagital, Coronal, and Turning). Furthermore, Matsuzawa developed independent crawling motion behavior for moving on rough terrain [132]. Brandao et al proposed footstep planning for slippery and slanted terrain [28]. In WAREC-1 robot, trajectory based locomotion generator also implemented, they developed different motion behavior such as: bipedal/quadrupedal walking, crawling, and ladder climbing [70]. In order to the footstep movement, A* algorithm is implemented. However, those conventional model have some limitation and inefficient in dynamic locomotion planning. They have to model certain locomotion planning in order to generate certain motion

behavior. These motion strategy will receive no problems when implements small number of behavior and implemented in non-global area. When we implement in global area, number of motion behavior will be increased. These implies requirement to model many number of behavior and requires a high number of memory and computational cost. These kind of problems is still becoming a big issues and makes saturation development in conventional model. Therefore, we tried to implemented neural based model as alternate way instead of conventional model. This model uses natural process and has higher expectation in the future. In order to simplify the technical background of neural based motion capabilities model, we separated into three part, locomotion behavior, stability model, and motion planning model.

#### 1.1.2.1 Locomotion behavior

From explanation above, the conventional model does not represent well natural behavior during locomotion. Since the trajectory based locomotion model generate in cartesian level, it requires inverse kinematic model for converting into joint angle level. It also needs high mathematical complexity to realize the dynamic walking pattern. In other hand, bio-inspired model may be the alternate approach for considering the dynamical system, non-linear system, and natural process. Bio-inspired locomotion provides bottom-up process, where the joint angles are firstly generated before trajectory in Cartesian level. In neural oscillator model provide simple solution which can be applied in complex problem, because its non-linearity. Therefore, this model can decrease the computational cost. Since 2000, bio-inspired locomotion developments are exponentially increased. Although the performance results are not better than conventional approach, bio-inspired model have higher expectation performance than conventional approach because it proved non linearity and dynamism.

The controller system in the brain has been proposed by Roy [164]: this is the basis of how the brain controls locomotion. Locomotion models based on a biological approach have been proposed by several researchers [86, 84, 196, 136, 142]. Before we applied locomotion in a multi-legged robot based on the neuro-biological approach, we studied the locomotion system as adjusted by animal morphologies [51]. Four-legged animal locomotion has been proposed by Ijspeert et al. They control animal locomotion by using a neural oscillator and also design the transition mechanism from swimming to walking [86], [84], [85]. Locomotion based on the central pattern generation (CPG) approach in four-legged animal robots has been applied by several researchers [51]. Furthermore, CPG can also be applied for malfunction compensation in six-legged robots [158].

A neural oscillator is also implemented for legged robot locomotion, as proposed by several researchers [196, 136, 142, 88, 14]. Taga et al. used coupled neurons for generating the

oscillated signal to drive the joint. They dealt with a sensory feedback system to adapt to the environmental conditions and created a mathematical model in order to acquire the feedback calculation. Their proposed method is applied in computer simulation [196]. Another neuro-model of locomotion is presented by Matos et al., who proposed a CPG approach based on phase oscillators for bipedal locomotion [127]. However, the ability to recover the disturbance is required. Ishiguro et al. also proposed the concept of a neural oscillator to realize two-legged robot locomotion. This model is applied to control a three-dimensional biped robot that is intrinsically unstable. They applied a feedback sensor to form dynamic locomotion; however, the robot has a limited degree of freedom and is applied at simulation level only [88]. In 2014, Nassour et al. proposed the locomotion model in the humanoid biped robot: they extended the mathematical model of CPG and designed a multi-layered neuron connection in order to control various models of walking [142], [141]. Nassour's research has good stability; however, the aim of our research is to improve the stability level of walking. In 2010, Park et al. designed a locomotion using an evolutionary optimized CPG. They also proposed sensory feedback to support the walking model; however, they did not consider a learning system for stability [148]. Other researchers have considered center of mass (COM) for their locomotion, based on central pattern generation [79], [149]. They have not considered however the stability of the learning system, or control to get various walking patterns.

In this current issue of bio-inspired model, stability, walking provision (omni-directional walking), and learning process are obstacle in this locomotion development. Omni-directional walking are solved in this proposed research. Omni directional locomotion model provides dynamic movement and ability to modify its motion quickly so that support the robot to move in dynamic environment [12].

Most of the limit cycle development only consider speed in unidirectional walking [77, 52, 68]. Endo *et al* developed adjusted walking velocity for neural oscillator based locomotion. However, range of adjusted velocity is small [52]. In 2008, Manoonpong *et al* developed neural based locomotion in order to generate omnidirectional movement in any types of legged robot. The proposed model can easily be adapted to control other kinds of walking machine without changing the internal network structure and its parameters. This model also can produce at least 11 different walking patterns and a self-protective reflex by using five input neurons [124]. Therefore in this thesis, we proposed dynamic structure model in neural oscillator based locomotion and learning model in order to generate unscaled omnidirectional movement in biped robot for solving current issues in neural based locomotion model.

#### 1.1.2.2 Stability behavior

Stability is the important part in bipedal humanoid robot. Robots have to walk stably in any condition and have to be ready getting external disturbance or internal disturbance. Following the humanoid robot development, the size of its footprint is getting smaller. It causes the walking of humanoid robot mainly not stable and its stability is difficult to be obtained. In the current state of the stability development, open loop based control is the most famous stability model applied in humanoid robot, especially in small humanoid robot. It results a walking model with good stability. However, pre-learning processes on well-defined surface are required. By using this technique, robot will be unstable when it finds undefined surface and undefined disturbance. Therefore, online stability learning model may be a great model in order to acquire a good stability [230].

In online based stability model, most researchers implemented physical method for the stabilization. They implemented inertial sensor or physical sensor and calculated the torques required in each joint for responding to any external disturbances. [82, 81] We also implemented physical approach in previous research. We applied inverted pendulum model and zero moment point in humanoid robot locomotion [176], [169]. On the other hand, researchers use biomechanical approach, applied the human behavior in order to recover and reach the stability level. Human shows the three different motion behaviors for responding sudden external disturbances, which are ankle recovery, hip recovery, and step recovery strategy [9, 186]. This algorithm is claimed that it has lower computational cost than physical based model because of its simplicity. Yi et al applied push recovery controller integrated with three motion behaviors. However, these algorithms required a lot of training data [230, 231]. They also did not consider the energy required in stability activity that considered in this proposed research. Pre-defining the strategy classification is required to be performed before learning process, it is seemed as unnatural process in human behavior. [230] also has not been proved yet to be applied in humanoid robot that has human-like footprint. Our proposed model is therefore applied in human-like footprint or small footprint.

Furthermore, bio-inspired control system may be able to become a new innovation in push recovery controller. This algorithm shows the natural process of human model. Its effectiveness has been proved by several researchers [236], [62], [95]. Zhang et al shows the effectiveness of Recurrent neural network (RNN) as predictive control [236]. RNN was applied to control the stability of biped robot locomotion [115]. Neural network is also implemented in order to process sensory feedback information in central pattern generation (CPG) based locomotion [62]. Fukuda et al. combine RNN with evolutionary algorithm to achieve the stable locomotion in humanoid robot [59].

Depending on the current issues, in this proposed model, we applied bio-inspired stability controller for humanoid robot locomotion. This proposed stability model used multimodal learning system which can assume different condition. It assumes that robot performs different behavior in different condition. In the humanoid robot case, if the robot gets a small disturbance such as the push or uneven terrain, then the robot only gives hands response for protecting its stable condition. Three motion behaviors in biomechanical approach is also considered in this proposed model.

#### 1.1.2.3 Motion planning

During a disaster, the terrain, route, area, and also building are becoming unstructured, diverse, and challenging. It makes the rescuing process difficult and challenging for human or robot who is in charge in disaster area. The route in disaster area becomes unstructured, making it difficult for the rescuer whether human or robot to find the best and possible route in the rough and unstructured terrain. Since it is very dangerous for human to rescue in unstable environment, robot is the best choice for exploring and investigating the disaster area. Most researchers proposed wheeled mobile robot for rescuing in disaster areas [139, 159, 182], but legged robot is more effective in rough terrain [11]. Therefore, we applied this proposed algorithm in a four-legged robot which is equipped with several supporting sensors. Four-legged robots are more efficient than biped robots in the stabilization cases and are able to achieve the maximum movement speed compared to robots with more legs. Furthermore, 3-D path planning model is required in order to support and facilitate those robots while moving on rough terrain. Therefore, in this research, we propose an online 3-D path planning optimization based on neural activity.

Three dimensional path planning studies have used images for path planning and also applied multiclass support vector machines for obstacle avoidance [137]. By using this idea, the robot can generate the safe pathway. In [179], 3-D based path planning was proposed, and D* algorithm was modified in order to estimate the distance in sloping terrain. Beside that, Dogru et al proposed genetic algorithm for optimizing pathway with minimum energy required [43, 44]. However, [179, 43, 44] did not consider the unpredictable obstacle. Beside that, Kroumov et al solved the obstacle problem but there is still problem with concave 3-D obstacles [103]. 3-D path planning was also proposed for UAV movement algorithm [145, 166, 211]. These algorithms were applied in computer simulation. In the integration system, UAV or drone is used for generating the map. Some researchers used 3-D map reconstruction in order to acquire the 3-D map model and to find the best pathway based on the result of reconstruction map. The robot was equipped with laser range finder (LRF) or

Kinect sensor in order to support the reconstruction algorithm. [26] also used LRF to generate the 2-D maps. This idea deals with static environment. Henry et al used Kinect camera in order to model 3-D indoor environment [74]. In the previous research, multi-resolution map was also used for decreasing the computational cost existing in high resolution map of path planning [201]. After that, a real-time feature extraction and segmentation method for a 3-D map [202] was proposed in order to increase the efficiency of the topological map. This map model can decrease the memory usage for reconstructing the map. Therefore, [202] is suitable to be combined for the proposed 3-D path planning model in the next stage.

In further cases of the environment model generated by LRF and Kinect sensor, the unpredictable surface such as friction, unstable terrain, and unpredictable collision sometimes become the problem. Those sensors are used as initial data in path planning, therefore those ones only deal with static path planning. In the real cases, the robot deals with dynamic environment, and it should autonomously work through dynamic environment. When the algorithm deals only with static environment, the robot will get problem when it finds an unpredictable collision. Therefore, the dynamic path planning is important in order to deal with dynamic environment. Dynamic path planning will regenerate the path planning when the algorithm finds unpredictable condition or changing environmental condition. In order to support the dynamic path planning, either additional sensors in mobile robot or measurement tools for human are required to measure and detect unpredictable collisions (friction, obstacle, and unstable of terrain) which cannot be considered in initial map generated by LRF or Kinect sensor.

Most mobile robots (wheeled and legged) are supplemented with the capability for finding the pathway. Some people used traditional algorithm such as: A*, D*, and Dijkstra algorithm in order to find the best pathway [114], [53], [143], [42], [113]. Ferguson et al proposed a modified D* algorithm in order to reduce the path cost in non-uniform environment. An interpolation equation was proposed in order to cut the inefficient pathway [53]. However, in [53]'s algorithm more computational cost was required. In common cases, D* path planning algorithm is faster than A*, but [75] modified the adaptive A*, therefore it could be faster than D* in some cases. These algorithms [75, 53] required predefined travel cost, therefore these algorithms seem difficult to cover unpredictable obstacles.

However, the traditional algorithm of path planning requires the determination of the path planning rule and it uses a recursive algorithm. Therefore, these algorithms require high computational cost. Bio-inspired algorithm provides a natural process that is able to dynamically generate the best pathway [160, 154]. Some researchers have proved the effectiveness of neural based model for path planning problems [64], [224], [227], [154]. In [64], Hopfield method was applied in a neural network to generate the pathway with obstacle avoidance.

This proposed model was applied in a 2-dimensional simulation. Neural based approach can also be applied for Complete Coverage Path Planning with obstacle avoidance [224]. Quoy et al proposed the control model in mobile robot by using dynamical neural networks based on the neural field formalism that was applied in the mobile robot path planner [154]. Most of the researchers applied pulsed neural network in order to find the efficient pathway [152], [153], [238, 222]. They focused on the inner activity of the neuron. Qu et al proposed pulsed neural network to generate real time collision free path planning [152]. Zhang et al applied a simple pulse coupled neural network to decrease the computational cost, but the defined travel cost is required in this case [238].

Furthermore, modified pulsed neural networks were proposed by several researchers in order to increase the performance of the path planning algorithm [111, 67, 153]. In [67], quick optimization process of path planning was proposed by using PCNN model. However, there is no efficient model to reduce the number of evolved neurons, there are many neurons required for representing the best path way. [235] presents a coupled neural network, called output-threshold coupled neural network (OTCNN). However, this algorithm was very hard to be implemented in the real cases and it was improved in [153] by proposing a new modified model of Pulse-coupled neural networks (M-PCNNs). This model was improved by Liu et al by solving the K Shortest Paths (KSPs) problem [119]. In addition, the neural network based path planner can also be applied in non-holonomic mobile robot [227]. Furthermore, shunting based neural model first proposed by Hodgkin and Huxley [78] and refined by Grossberg in the neural mechanism [66] was also used to solve path planning problems [226, 225]. However, the current neural based models [238, 226, 225, 119, 227, 111] have not considered rough terrain with undefined travel cost or weighting cost.

## 1.2 Motivation and objectives

The main purposed of this theses is for developing motion capabilities of multi-legged robot using neural based model as the alternate way for avoiding the saturation issues in conventional model. There are three part of motion capabilities should be developed which are locomotion behavior, stability model, and motion planning model. The main work and contribution of this theses are following:

- Locomotion behavior
  - Using musculoskeletal model as the locomotion model in multi-legged robot where one joint is inspired by coupled neurons representing flexor and extensor muscle

  - Developing neural oscillator interconnection model which can generate omni-directional motion behavior.
  - Creating online learning model for generating interconnection structure of motor neurons and sensory-motor neuron using evolutionary algorithm

- Stability model
  - Designing multimodal learning system for stability which implement combined Recurrent NNs (MRNN), which work alternately depending on the current condition.
  - Using natural process for push recovery behavior controller, where one MRNN representing one motion behavior.
  - Modifying Multivariate adaptive regression splines (MARS) as online learning model for selecting appropriate behavior

- Motion planning
  - Using natural mechanism of the human brain for generating the online path planning in 3-D rough terrain with unpredictable travel cost.
  - Developing forward transmission to construct the neuron connections for finding the possible way and the backward neuron transmission with synaptic pruning model for finding the best pathway from the current position to the target position and for reducing inefficient neurons.

## 1.3 Thesis structure

Below, we outline the remainder of the thesis. We are going to discuss in deep the elements that built this thesis, starting from several explanation of multi-legged robot system until discussing our proposed model. The main of proposed model will be shown in $Chapter5$, 6, 7. It shows several model with different ideas where its experimental result will be shown in every models. The thesis structure can be seen in Fig. 1.2 and the detail outline of the remaining chapters is as follows:

- $Chapter2$ : We discuss several developement of multi-legged robot and the systems which support a motion capabilities of the multi-legged robot

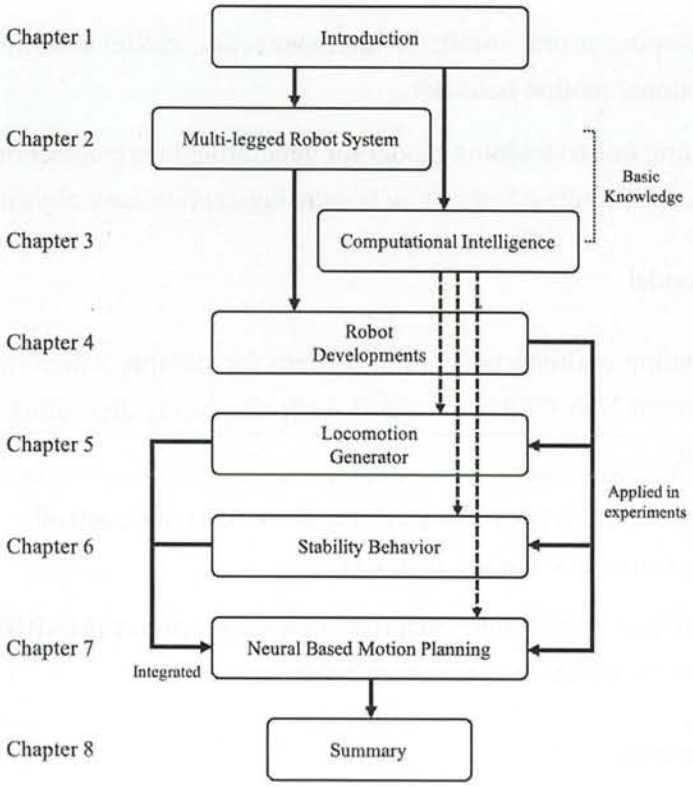



**Figure 1.2:** *The organization of the thesis*

- *Chapter*3 : We explain basic of computational intelligent and natural computing used in our proposed model. Fuzzy logic, neuro-based learning model, and also evolutionary algorithm will be shown.

- *Chapter*4 : We show several legged robot developments that used as object experiments of the proposed model. We also show several development of motion capabilities in biped robot using conventional model, in order to compare with proposed model. Experimental result will be shown in every development.

- *Chapter*5 : We present and discuss the proposed locomotion generator. Several system model developments in different ideas will be shown. The model improves the previous model development, such as: from single objective evolutionary algorithm usage to multi-objectives usage. Experimental result and discussion will be shown in every development.

- *Chapter*6 : We explain several developments of proposed stability model. Several implementations of proposed model will also be shown. It show the implementation in wheeled robot before implemented into legged robot.

- *Chapter*7 : We shows the proposed motion planning model based on natural processes. Integration with locomotion and stability model is also developed in order to show the experiment using legged robot.

- *Chapter*8 : We summarize our results and present future work of the motion capabilities of legged robot.

# Chapter 2

# Multi-legged Robot System

## 2.1 Multi-Legged Robot

### 2.1.1 A Brief History of Legged robot

The development of legged robot were around 1970 by Kato and Vukobratovic. Kato and his team demonstrated first antropomorphic robot called "WABOT1" in Waseda University Japan. The robot could realize several slow steps in static terrain using simple control model. It consisted of a limb-control system, a vision system and a conversation system. [8, 240]. In the same time, Vukobratovic and his team developed first active exoskeletons, and several other devices. Even thought their main purposes are in the problems generated by functional rehabilitation, however the most well-known outcome of their research is in analysis of locomotion stability, namely concept of zero moment point (ZMP) in 1972 [206]. This was the first attempt to formalize the need for dynamical stability of legged robots; the idea was to use the dynamic wrench in order to extend a classical criterion of static balance (the center of mass should project inside the convex hull of contact points).[91]. Furthermore, McGhee et al developed the gaits of six-legged robot called creeping gaits, seem to be well suited for low-speed locomotion since they permit a quadruped to remain statically stable during most of a locomotion cycle [135].

In the next decade, WABOT-1 was improved becoming WABOT-2. The purpose of this development was to realize 'soft' functions of robots such as dexterity, speediness and intelligence by the development of an anthropomorphic intelligent robot playing keyboard instrument.. Therefore the WABOT-2 was defined as a "specialist robot" rather than a versatile robot like the WABOT-1 [188]. Based on the previous development, R. McGhee and K. Waldron developed Adaptive Suspension Vehicle, which used a legged, locomotion principle, and is intended to demonstrate the feasibility of systems of this type for transportation

in very rough terrain conditions [208]. In 1985, Hitachi Ltd, in collaboration with Waseda University, developed WHL-11 (Waseda Hitachi Leg 11) a robot able to walk statically on a flat surface at 13 seconds per step and to turn [108].

In 1990s, McGeer introduced passive dynamic walking, as the invention of walking model, for a class of very simple systems: a plane compass on an inclined plane. Stable walking results from the balance between increase of the energy due to the slope and loss at the im- pacts. However, what should be emphasized here is that McGeer popularized for roboticists the analysis of such systems in terms of orbital stability using Poincare maps. [134]. Several researchers have followed the tracks open by McGeer, with many extensions: adding trunk, feet and knees [217], semipassive control, walking/running underactuated systems like the Rabbit robot [34], etc.

In the millennial era, there are a lot of legged robots have been developed. Industrial breakthroughs supported the development of real legged robot, so that the real humanoid was now possible. In 1996, first performance humanoid robot was exhibited by Honda. After that, impressive achievement in development of legged robot are realized by several industrial companies: ASIMO (Honda), QRIO (Sony), HRP (Kawada). Finally, in 21st century, there are a lot of powerful legged robot developed. Boston Dynamic currently is developing several legged robot, such as quadruped robot, biped robot, etc. One of them called BigDog, has four legs that are articulated like an animal's, with compliant elements to absorb shock and recycle energy from one step to the next. BigDog is the size of a large dog or small mule. It equipped by several intelligential sensors [2].

### 2.1.2 Type of legged robot

Legged robots can be categorized by the number of limbs they use, which determines gaits available. Many-legged robots tend to be more stable, while fewer legs lends itself to greater maneuverability.

#### 2.1.2.1 One-legged robot

One-legged, or pogo stick robots use a hopping motion for navigation. In the 1980s, Carnegie Mellon University developed a one-legged robot to study balance.

#### 2.1.2.2 Two-legged robot

Bipedal or two-legged robots exhibit bipedal motion. As such, they face two primary problems: stability control, which refers to a robot's balance, and motion control, which

refers to a robot's ability to move. Stability control is particularly difficult for bipedal systems, which must maintain balance in the forward-backward direction even at rest. Some robots, especially toys, solve this problem with large feet, which provide greater stability while reducing mobility. Alternatively, more advanced systems use sensors such as accelerometers or gyroscopes to provide dynamic feedback in a fashion that approximates a human being's balance. Such sensors are also employed for motion control and walking. The complexity of these tasks lends itself to machine learning.

Simple bipedal motion can be approximated by a rolling polygon where the length of each side matches that of a single step. As the step length grows shorter, the number of sides increases and the motion approaches that of a circle. This connects bipedal motion to wheeled motion as a limit of stride length.

Two-legged robots include: Toy robots such as QRIO and ASIMO, NASA's Valkyrie robot, intended to aid humans on Mars, the ping-pong playing TOPIO robot, etc

#### 2.1.2.3 Four-legged robot

Quadrupedal or four-legged robots exhibit quadrupedal motion. They benefit from increased stability over bipedal robots, especially during movement. At slow speeds, a quadrupedal robot may move only one leg at a time, ensuring a stable tripod. Four-legged robots also benefit from a lower center of gravity than two-legged systems.

Four legged robots include: the TITAN series, developed since the 1980s by the Hirose-Yoneda Laboratory, the dynamically stable BigDog, developed in 2005 by Boston Dynamics, NASA's Jet Propulsion Laboratory, and the Harvard University Concord Field Station, BigDog's successor, the LS3, etc.

#### 2.1.2.4 Six-legged robot

Six-legged robots, or hexapods, are motivated by a desire for even greater stability than bipedal or quadrupedal robots. Their final designs often mimic the mechanics of insects, and their gaits may be categorized similarly. These include: (1) Wave gait: the slowest gait, in which pairs of legs move in a "wave" from the back to the front. (2) Tripod gait: a slightly faster step, in which three legs move at once. The remaining three legs provide a stable tripod for the robot.

Six-legged robots include: (1) Odex, a 375-pound hexapod developed by Odetics in the 1980s. Odex distinguished itself with its onboard computers, which controlled each leg. (2) Genghis, one of the earliest autonomous six-legged robots, was developed at MIT by Rodney Brooks in the 1980s. (3) The modern toy series, Hexbug.

#### 2.1.2.5 Eight-legged robot

Eight-legged legged robots are inspired by spiders and other arachnids, as well as some underwater walkers. They offer by far the greatest stability, which enabled some early successes with legged robots Eight-legged robots include: (1) Dante, a Carnegie Mellon University project designed to explore Mount Erebus. (2) The T8X, a commercially available robot designed to emulate a spider's appearance and movements.

#### 2.1.2.6 Hybrid robot

Some robots use a combination of legs and wheels. This grants a machine the speed and energy efficiency of wheeled locomotion as well as the mobility of legged navigation. Boston Dynamics' Handle, a bipedal robot with wheels on both legs, is one example.

## 2.2 Kinematic Model

Two main tasks are involved in the design of locomotion strategies for multi-legged robots walking over any terrain. One is the high level leg sequencing and motion planning. The other is lower level actuation and coordinated control of robot joints to achieve a desired motion of the rover body. While the high level task is crucial and a large volume of research has been devoted to it, the lower level task is also very important for the success of walking. This lower level task requires developing kinematics models for the multi-legged robot but has received little attention. Kinematic model will implies embodiment structure of the robot. That will effect to the motion behavior. Kinematic model composes two model, forward kinematic and inverse kinematic.

### 2.2.1 Forward Kinematics

Forward kinematics refers to the use of the kinematic equations of a robot to compute the position of the end-effector from specified values for the joint parameters [150]. The kinematics equations of the robot are used in robotics, computer games, and animation. The reverse process that computes the joint parameters that achieve a specified position of the end-effector is known as inverse kinematics.

The kinematics equations for the series chain of a robot are obtained using a rigid transformation $[Z]$ to characterize the relative movement allowed at each joint and separate rigid transformation $[X]$ to define the dimensions of each link. The result is a sequence of rigid

transformations alternating joint and link transformations from the base of the chain to its end link, which is equated to the specified position for the end link,

$$[T] = [Z_1][X_1][Z_2][X_2]\dots[Z_{n-1}][X_{n-1}] \tag{2.1}$$

where $[T]$ is the transformation locating the end-link. These equations are called the kinematics equations of the serial chain.

In 1955, Jacques Denavit and Richard Hartenberg introduced a convention for the definition of the joint matrices [Z] and link matrices [X] to standardize the coordinate frame for spatial linkages [40, 69]. This convention positions the joint frame so that it consists of a screw displacement along the Z-axis and it positions the link frame so it consists of a screw displacement along the X-axis.

$$[Z_i] = \text{Trans}_{z_i}(d_i)\text{Rot}_{z_i}(\theta_i) \tag{2.2}$$

$$[X_i] = \text{Trans}_{x_i}(r_{i,i+1})\text{Rot}_{x_i}(\alpha_{i,i+1}) \tag{2.3}$$

$$^{i-1}T_i = [Z_i][X_i] = \text{Trans}_{z_i}(d_i)\text{Rot}_{z_i}(\theta_i)\text{Trans}_{x_i}(r_{i,i+1})\text{Rot}_{x_i}(\alpha_{i,i+1}) \tag{2.4}$$

Using this notation, each transformation-link goes along a serial chain robot, and can be described by the coordinate transformation, where $\theta_i$, $d_i$, $r_{i,i+1}$ and $\alpha_{i,i+1}$ are known as the Denavit-Hartenberg parameters.

#### 2.2.1.1 Denavit-Hartenberg convention

A commonly used convention for selecting frames of reference in robotics applications is the Denavit and Hartenberg (D–H) convention which was introduced by Jacques Denavit and Richard S. Hartenberg. In this convention, coordinate frames are attached to the joints between two links such that one transformation is associated with the joint, $[Z]$, and the second is associated with the link $[X]$. The coordinate transformations along a serial robot consisting of n links form the kinematics equations of the robot in Eq. (2.1).

In order to determine the coordinate transformations $[Z]$ and $[X]$, the joints connecting the links are modeled as either hinged or sliding joints, each of which have a unique line S in space that forms the joint axis and define the relative movement of the two links. A typical serial robot is characterized by a sequence of six lines $Si$, $i = 1, \dots, 6$, one for each joint in the robot. For each sequence of lines $S_i$ and $S_{i+1}$, there is a common normal line $A_{i,i+1}$. The system of six joint axes $S_i$ and five common normal lines $A_{i,i+1}$ form the kinematic skeleton

of the typical six degree of freedom serial robot. Denavit and Hartenberg introduced the convention that Z coordinate axes are assigned to the joint axes Si and X coordinate axes are assigned to the common normals $A_{i,i+1}$.

This convention allows the definition of the movement of links around a common joint axis $S_i$ by the screw displacement,

$$[Z_i] = \begin{bmatrix} \cos\theta_i & -\sin\theta_i & 0 & 0 \\ \sin\theta_i & \cos\theta_i & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.5}$$

where $\theta_i$ is the rotation around and di is the slide along the Z axis—either of the parameters can be constants depending on the structure of the robot. Under this convention the dimensions of each link in the serial chain are defined by the screw displacement around the common normal $A_{i,i+1}$ from the joint $S_i$ to $S_{i+1}$, which is given by

$$[X_i] = \begin{bmatrix} 1 & 0 & 0 & r_{i,i+1} \\ 0 & \cos\alpha_{i,i+1} & -\sin\alpha_{i,i+1} & 0 \\ 0 & \sin\alpha_{i,i+1} & \cos\alpha_{i,i+1} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.6}$$

where $\alpha_{i,i+1}$ and $r_{i,i+1}$ define the physical dimensions of the link in terms of the angle measured around and distance measured along the X axis. In summary, the reference frames are laid out as follows:

1. the z-axis is in the direction of the joint axis
2. the x-axis is parallel to the common normal: $x_n = z_n \times z_{n+1}$. If there is no unique common normal (parallel z axes),
3. then d (below) is a free parameter. The direction of $x_n$ is from $z_{n-1}$ to $z_n$, as shown in the Fig. 2.1.

The following four transformation parameters are known as DH parameters: $d$: offset along previous z to the common normal; $\theta$ : angle about previous $z$, from old $x$ to new $x$; $r$ : length of the common normal. Assuming a revolute joint, this is the radius about previous z; $\alpha$ : angle about common normal, from old $z$ axis to new $z$ axis

There is some choice in frame layout as to whether the previous $x$ axis or the next $x$ points along the common normal. The latter system allows branching chains more efficiently, as

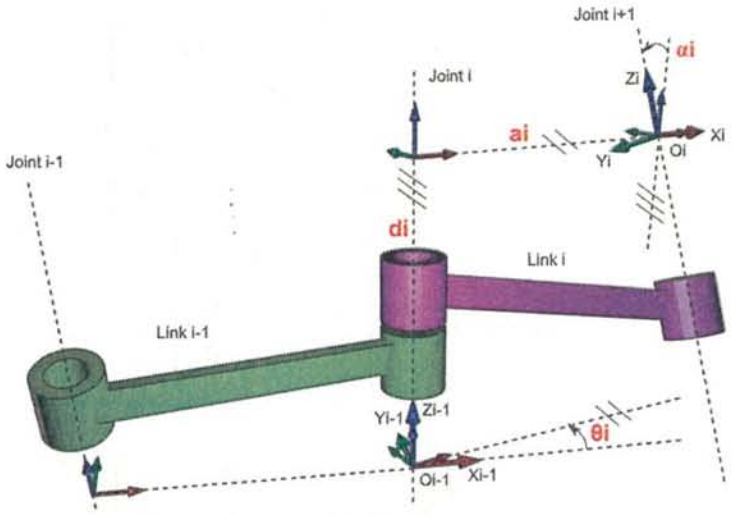



**Figure 2.1:** *The four parameters of classic DH convention are shown in red text, which are $\theta_i$,$d_i$,$a_i$,$\alpha_i$. With those four parameters, we can translate the coordinates from $O_{i-1}X_{i-1}Y_{i-1}Z_{i-1}$ to $O_iX_iY_iZ_i$ [3]*

multiple frames can all point away from their common ancestor, but in the alternative layout the ancestor can only point toward one successor. Thus the commonly used notation places each down-chain $x$ axis collinear with the common normal, yielding the transformation calculations shown below. We can note constraints on the relationships between the axes: (1) the $x_n$-axis is perpendicular to both the $z_{n-1}$ and $z_n$ axes; (2) the $x_n$-axis intersects both $z_{n-1}$ and $z_n$ axes; (3) the origin of joint $n$ is at the intersection of $x_n$ and $z_n$; (4) $y_n$ completes a right-handed reference frame based on $x_n$ and $z_n$.

It is common to separate a screw displacement into the product of a pure translation along a line and a pure rotation about the line,[5][6] so that $[Z_i] = \mathrm{Trans}_{Z_i}(d_i)\,\mathrm{Rot}_{Z_i}(\theta_i)$, and $[X_i] = \mathrm{Trans}_{X_i}(r_{i,i+1})\,\mathrm{Rot}_{X_i}(\alpha_{i,i+1})$. Using this notation, each link can be described by a coordinate transformation from the concurrent coordinate system to the previous coordinate system.

Note that this is the product of two screw displacements, The matrices associated with these operations are:

$$\mathrm{Trans}_{z_{n-1}}(d_n) = \left[\begin{array}{ccc|c} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_n \\ \hline 0 & 0 & 0 & 1 \end{array}\right] \tag{2.7}$$

$$\mathrm{Rot}_{z_{n-1}}(\theta_n) = \left[\begin{array}{ccc|c} \cos\theta_n & -\sin\theta_n & 0 & 0 \\ \sin\theta_n & \cos\theta_n & 0 & 0 \\ 0 & 0 & 1 & 0 \\ \hline 0 & 0 & 0 & 1 \end{array}\right] \tag{2.8}$$

$$\mathrm{Trans}_{x_n}(r_n) = \left[\begin{array}{ccc|c} 1 & 0 & 0 & r_n \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ \hline 0 & 0 & 0 & 1 \end{array}\right] \tag{2.9}$$

$$\mathrm{Rot}_{x_n}(\alpha_n) = \left[\begin{array}{ccc|c} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha_n & -\sin\alpha_n & 0 \\ 0 & \sin\alpha_n & \cos\alpha_n & 0 \\ \hline 0 & 0 & 0 & 1 \end{array}\right] \tag{2.10}$$

$${}^{n-1}\mathrm{T}_n = \left[\begin{array}{ccc|c} \cos\theta_n & -\sin\theta_n\cos\alpha_n & \sin\theta_n\sin\alpha_n & r_n\cos\theta_n \\ \sin\theta_n & \cos\theta_n\cos\alpha_n & -\cos\theta_n\sin\alpha_n & r_n\sin\theta_n \\ 0 & \sin\alpha_n & \cos\alpha_n & d_n \\ \hline 0 & 0 & 0 & 1 \end{array}\right] = \left[\begin{array}{ccc|c} & R & & T \\ \hline 0 & 0 & 0 & 1 \end{array}\right] \tag{2.11}$$

In Eq. (2.11), R is the 3×3 submatrix describing rotation and T is the 3×1 submatrix describing translation.

### 2.2.2 Inverse Kinematics

Inverse kinematics is the mathematical process of recovering the movements of an object in the world from some other data, such as a film of those movements, or a film of the world as seen by a camera which is itself making those movements. This is useful in robotics and in film animation. In legged robot, it converts from cartesian level to joint angle level.

There are many methods for solving inverse kinematic problems. Whitney el al proposed pseudo inverse method [216] and Wang et al proposed cyclic coordinate descent methods [212] in 1991. Furthermore, some of researcher also used Jacobian transpose methods [19, 218], quasi-Newton and conjugate gradient methods [239, 41], the Levenberg-Marquardt damped least squares methods [209, 140].

Currently, most researchers implemented Jacobian model for solving IK problems. The Jacobian inverse technique is a simple yet effective way of implementing inverse kinemat-

ics. Let there be $m$ variables that govern the forward-kinematics equation, $i.e.$ the position function. These variables may be joint angles, lengths, or other arbitrary real values. If the IK system lives in a 3-dimensional space, the position function can be viewed as a mapping $p(x) : \Re^m \rightarrow \Re^3$. Let $p_0 = p(x_0)$ give the initial position of the system, and $p_1 = p(x_0 + \Delta x)$ be the goal position of the system. The Jacobian inverse technique iteratively computes an estimate of $\Delta x$ that minimizes the error given by $||p(x_0 + \Delta x) - p_0||$.

For small $\Delta x$-vectors, the series expansion of the position function gives: $p(x_1) \approx p(x_0) + J_p(x_0)\Delta x$, where $J_p(x_0)$ is the (3 x $m$) Jacobian matrix of the position function at $x_0$.

Note that the $(i, k)$-th entry of the Jacobian matrix can be determined numerically: $\frac{\partial p_i}{\partial x_k} \approx \frac{p_i(x_{0,k}+h)-p_i(x_0)}{h}$. Where $p_i(x)$ gives the $i$-th component of the position function, $x_{0,k} + h$ is simply $x_0$ with a small delta added to its $k$-th component, and $h$ is a reasonably small positive value. Taking the Moore-Penrose pseudoinverse of the Jacobian (computable using a singular value decomposition) and re-arranging terms results in: $\Delta x \approx J_p^+(x_0)\Delta p$, where $\Delta p = p(x_0 + \Delta x) - p(x_0)$.

Applying the inverse Jacobian method once will result in a very rough estimate of the desired $\Delta x$-vector. A line search should be used to scale this $\Delta x$ to an acceptable value. The estimate for $\Delta x$ can be improved via the following algorithm (known as the Newton-Raphson method): $\Delta x_{k+1} = J_p^+(x_k)\Delta p_k$. Once some $\Delta x$-vector has caused the error to drop close to zero, the algorithm should terminate. Existing methods based on the Hessian matrix of the system have been reported to converge to desired $\Delta x$ values using fewer iterations, though, in some cases more computational resources.

In order to simplify the equation, some researcher only used trigonometry based for solving the inverse kinematic problem. However, this trick has limitation in mechanical structure and number of joints.

In different way, some researchers propose neural model and artificial intelligent based model [147, 156, 198, 46, 49, 100, 101]. Duka et al used neural network with fitting problem for solving the IK. The inverse kinematics problem for a three-link robotic manipulator was transformed in a fitting problem, in which a neural network was used to map between a data set of numeric inputs and a set of numeric targets [49]. He also developed Adaptive Neuro-Fuzzy Inference System (ANFIS) based for IK solution. ANFIS is trained using the inverse kinematic mapping of the data provided by forward kinematics and learns, with acceptable accuracy, the end-effector's localization to joint angle mapping [50]. Almusawi et al proposed ANN with feedback of current joint angles configuration of robotic arm in the input point of their network [13].

Koker et al developed a neural network for planar robotic manipulator [101]. He tried to

improve the preciseness of the neural-network-based inverse kinematics solution using a genetic algorithm [100]. In 2010, Bouganis et al developed the control model for 4-DoF robotic arm suing spiking neural network. The trained spiking neural network has been successfully tested on a kinematic model of the arm of an iCub humanoid robot. However, the current model of the network is computationally expensive (several seconds of processing time are required per arm movement) [27].

## 2.3 Motion capabilities

Such as in human capabilities, motion capabilities is responsible for performing physical activity such as 1) motion behavior, how the human generate different behavior of movement in different condition, and 2) stability behavior, how the human respond the disturbance. Appropriate behavior should be generated in certain environmental condition. Some of researcher also put its motion planning as the content of motion capabilities. In motion planning, the human know how to move from one place to another place. In order to move, robot should know the way to the goal point. Robot have to know how to move through the way with uncertain condition, and how to stabilize from the disturbance. Those systems are important systems should be integrated in order to build motion capabilities model. Therefore, in this chapter we separated some basic theories into those three parts.

### 2.3.1 Locomotion Generator

Robotic locomotion is one of the important functions of the robotic device that helps the platform in traversing rough terrain; moving and interacting in human environment. Although wheeled locomotion is very common, foolproof, energy efficient and easily controllable; there are other motion alternativesare also available such as legged motion including Bipedal, Quadrapedal, Hexapedal, and soon. Track belt, walking, running, hopping, swimming, slithering, brachiating, etc are the other behavior forms used in locomotion behavior.

Some researcher used nature principle as the basic of locomotion which can be seen in Fig. 2.2. In this part, we separate the locomotion model into 3 parts, which are, trajectory based locomotion, limit cycle based locomotion, and passive locomotion.

#### 2.3.1.1 Trajectory based Locomotion

Trajectory based locomotion of legged robot generates the point of movement of end effection in each limb of legged robot in cartesian level. It requires inverse kinematic for

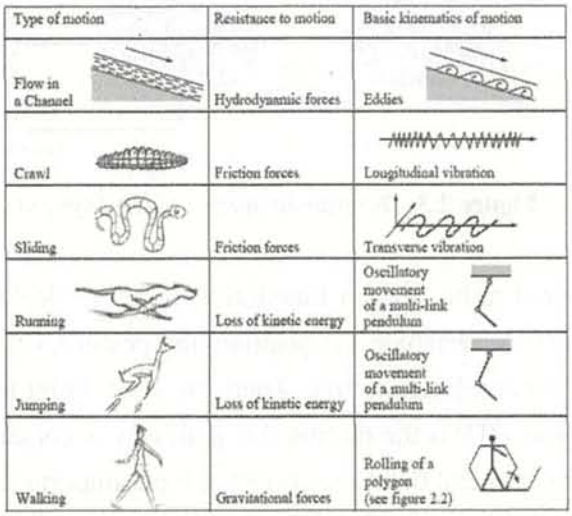



**Figure 2.2:** *Nature based locomotion principles [5]*

converting from cartesian level to actuator level, (joint angle or slide value). In the current development, there are a lot of legged robot development implementing conventional model of locomotion or trajectory based locomotion. HUBO [237, 210], KHR [97, 96], BigDog [155, 221], HyQ [181], ATLAS [105, 197, 37, 106], HRP [232], etc implemented trajectory based locomotion model for their motion planning. ATLAS implemented ZMP based locomotion model that used ZMP as the variable to be controlled. [105, 197, 37, 106].

In legged robot, BigDog walks with a dynamically balanced trot gait. It balances using an estimate lateral velocity and acceleration, determined from the sensed behavior the legs during stance combined with the inertial sensors. BigDog's control system coordinates the kinematics and ground reaction forces of the robot while responding to basic postural commands. The control distributes load amongst the legs to optimize their load carrying ability. The vertical loading across the limbs is kept as equal as possible while individual legs are encouraged to generate ground reactions directed toward the hips, thus lowering required joint torques and actuator efforts [155, 221]. Basic walking control uses the control system diagrammed below. A gait coordination algorithm, responsible for inter- leg communication, initiates leg state transitions to produce a stable gait. A virtual leg model coordinates the legs.

Fig. 2.3 shows a framework of motion control system including the trajectory generation. Motion Planning is sometimes called Trajectory Planning. Motion Planning is the function in order to decide how to move based on intent of an operator or decision of a robot. If a control system is the position control system, the information that Motion Planning output is the position information; if a control system is the force control system, the force information is given. In the following, we give explanation on the premise of the position control. Generally,

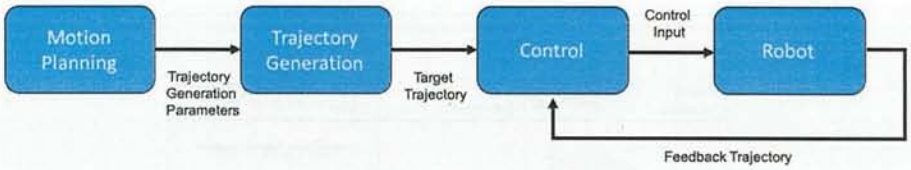



**Figure 2.3:** *Diagram of motion control system*

target trajectory is desired to be smooth function of time. To decide the "Specification of motion" is setting following variables, (1) position and posture, (2) maximum velocity, (3) maximum acceleration or acceleration time. There are PTP (Point to Point) motion and CP (Contineous Path) motion. PTP is the motion that path way is not considered if the position and posture of beginning and end point are correct. CP put importance on path way too [61].

**2.3.1.1.1 Basic Orbital Function** In order to follow the target trajectory, a function of time form basis of the desired motion needs to be designed. A lot of methods to design a function of time has been proposed. Here, we give following two examples.

1. PTP motion: each drive axis is controlled so that it can reach a desired position and posture at the beginning and end of the motion.

2. CP motion: each drive axis is controlled so that it tracks the target trajectory designed to satisfy the specification of acceleration and deceleration. Specification of motion is designed from position and posture; the position is based on movement distance and the posture is based on equivalent angle.

**2.3.1.1.2 Design of Orbital Function Using Five-Dimensional Prenominal** The planning method of position, velocity, and acceleration by using $n$-dimensional is introduced. In this subsection, the polynomial means a function of time expressed as Eq.( 2.12).

$$\eta(t) = a_n t^n + a_{n-1} t^{n-1} + \cdots + a_2 t^2 + a_1 t + a_0 \tag{2.12}$$

where $n$ is the dimension of the polynomial, normally counting number. Coefficients $a_j (j = 0, 1, 2, \ldots, n)$ are decided by initial condition or termination condition. Assuming that Eq. (2.12) is the function of position, velocity and acceleration are first and second-order differential respectively. By differentiating Eq. (2.12), following equations are derived.

$$\dot{\eta}(t) = n a_n t^{n-1} + (n-1) a_{n-1} t^{n-2} + \cdots + 2 a_2 t + a_1 \tag{2.13}$$

$$\ddot{\eta}(t) = n(n-1) a_n t^{n-2} + (n-1)(n-2) a_{n-1} t^{n-3} + \cdots + 2 a_2 \tag{2.14}$$

Design procedure when $n = 5$ is shown here. Eq. (2.15) is the orbital function of position, and Eq. (2.16) and Eq. (2.17) are velocity and acceleration function respectively.

$$\eta(t) = a_5 t^5 + a_4 t^4 + a_3 t^3 + a_2 t^2 + a_1 t + a_0 \tag{2.15}$$

$$\dot{\eta}(t) = 5a_5 t^4 + 4a_4 t^3 + 3a_3 t^2 + 2a_2 t + a_1 \tag{2.16}$$

$$\ddot{\eta}(t) = 20a_5 t^3 + 12a_4 t^2 + 6a_3 t + 2a_2 \tag{2.17}$$

The coefficients are decided if 6 independent conditions such as initial and termination position, velocity, and acceleration, because unknown variables are 6. By using initial and termination time $t_0$ and $t_1$, initial and termination position, velocity and acceleration $\eta_0, \dot{\eta}_0, \ddot{\eta}_0, \eta_1, \dot{\eta}_1$, and $\ddot{\eta}_1$, the following equation is derived:

$$\begin{bmatrix} \eta_0 \\ \dot{\eta}_0 \\ \ddot{\eta}_0 \\ \eta_1 \\ \dot{\eta}_1 \\ \ddot{\eta}_1 \end{bmatrix} = \begin{bmatrix} t_0^5 & t_0^4 & t_0^3 & t_0^2 & t_0 & 1 \\ 5t_0^4 & 4t_0^3 & 3t_0^2 & 2t_0 & 1 & 0 \\ 20t_0^3 & 12t_0^2 & 6t_0 & 2 & 0 & 0 \\ t_1^5 & t_1^4 & t_1^3 & t_1^2 & t_1 & 1 \\ 5t_1^4 & 4t_1^3 & 3t_1^2 & 2t_1 & 1 & 0 \\ 20t_1^3 & 12t_1^2 & 6t_1 & 2 & 0 & 0 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \end{bmatrix} \tag{2.18}$$

From Eq. (2.18), if coefficients matrix is holomorphic, $a_1 - a_5$ are decided uniquely:

$$\begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \end{bmatrix} = \begin{bmatrix} t_0^5 & t_0^4 & t_0^3 & t_0^2 & t_0 & 1 \\ 5t_0^4 & 4t_0^3 & 3t_0^2 & 2t_0 & 1 & 0 \\ 20t_0^3 & 12t_0^2 & 6t_0 & 2 & 0 & 0 \\ t_1^5 & t_1^4 & t_1^3 & t_1^2 & t_1 & 1 \\ 5t_1^4 & 4t_1^3 & 3t_1^2 & 2t_1 & 1 & 0 \\ 20t_1^3 & 12t_1^2 & 6t_1 & 2 & 0 & 0 \end{bmatrix}^{-1} \begin{bmatrix} \eta_0 \\ \dot{\eta}_0 \\ \ddot{\eta}_0 \\ \eta_1 \\ \dot{\eta}_1 \\ \ddot{\eta}_1 \end{bmatrix} \tag{2.19}$$

#### 2.3.1.2 Limit Cycle Locomotion [61]

In mathematics, in the study of dynamical systems with two-dimensional phase space, a limit cycle is a closed trajectory in phase space having the property that at least one other trajectory spirals into it either as time approaches infinity or as time approaches negative infinity. Such behavior is exhibited in some nonlinear systems [24]. Limit cycles have been used to model the behavior of a great many real world oscillatory systems. The study of limit cycles was initiated by Henri Poincare.

Normally, the robot locomotion is expressed as a nonlinear system. Limit cycle, which is

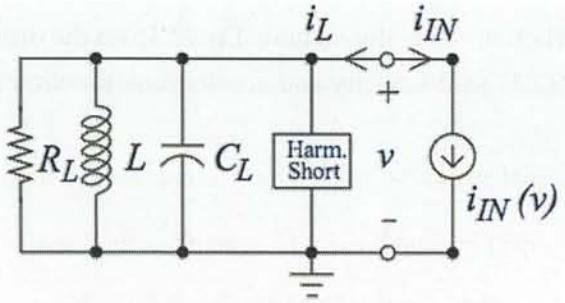



**Figure 2.4:** *Van Der Pol oscillator model*

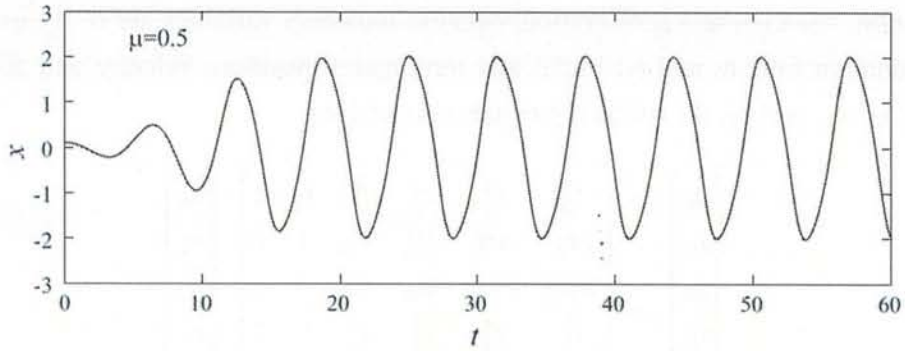



**Figure 2.5:** *Limit Cycle*

a phenomenon of nonlinear system, is often used for stability analysis. Limit cycle is the phenomenon that periodical solution with a constant period, amplitude, and angular frequency is generated out of relation to and initial value. Representative example is the van der Pol oscillator Fig. 2.4. The dynamic characteristic is expressed as follows:

$$\ddot{x} - \mu(1 - x^2)\dot{x} + x = 0(\mu > 0) \tag{2.20}$$

When $x$ is small value, the damping term is negative value; thus the system is unstable. In contrast, if the x is larger, the damping term is positive value; then the system will be stable and the oscillation will be steady at some value of $x$. Fig. 2.5 shows an example of $x$ behavior. The $x$ behavior of this example depicted in the $x`\dot{x}$ plain as shown in Fig. 2.6. The $x`\dot{x}$ plain is called "phase plane". To analyze behavior of solution trajectory in phase plane is called "phase plane analysis". The system needs to be an autonomous system when the phase plane is conducted.

In human like locomotion, the spine is an important part as in the human body. The flexibility of the human spine is much higher and that is very difficult to achieve artificially [47].

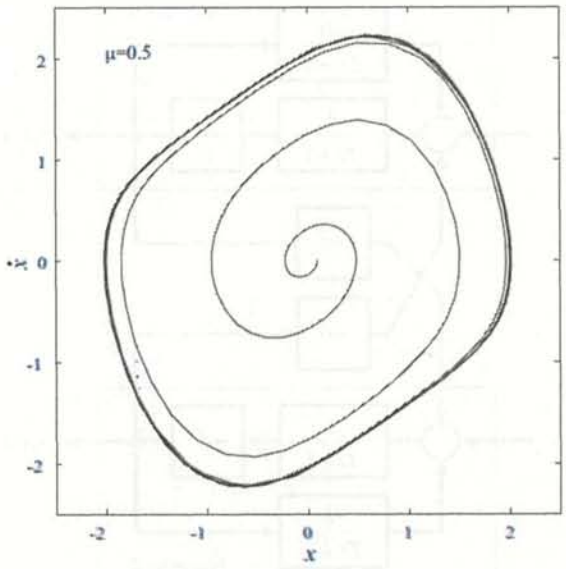



**Figure 2.6:** *$x$ behavior of Fig.2.5 in the $x - \dot{x}$ plain. (Limit Cycle)*

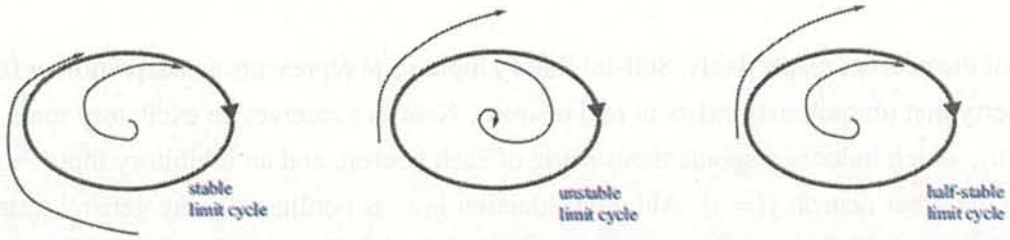



**Figure 2.7:** *Limit cycle stability*

**2.3.1.2.1 Matsuoka Model** The neural oscillator model proposed by Matsuoka [128, 130, 131] is composed of two identical neurons. The neuron model is an analog model; its output is not a train of spikes, but is a firing rate or a short time average of spike activity. The dynamics of each neuron $i(= 1, 2)$ is given by the following second-order system of differential equations, so that the total system is a fourth-order one:

$$\tau \frac{d}{dt} x_i(t) + x_i(t) = c - a y_j(t) - b_i(t), (i, j = 1, 2; j = i) \tag{2.21}$$

$$T \frac{d}{dt} v_i(t) + v_i(t) = y_i(t), \tag{2.22}$$

$$y_i(t) = g(x_i(t)) \tag{2.23}$$

Function $g(\cdot)$ is a piecewise linear function defined by $g(x) = max(0, x)$, which represents a threshold property of the neurons. The block diagram of the system is shown in Fig. 2.8. Variables $x_i(t)$ and $y_i(t)$ represent the so-called membrane potential and the firing

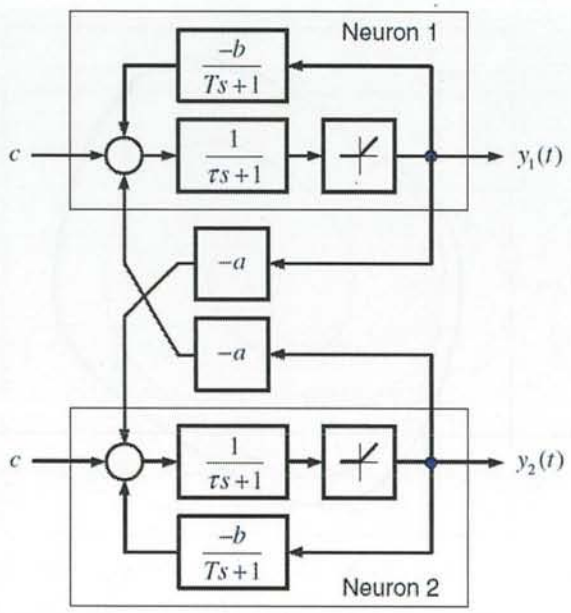



**Figure 2.8:** *The block diagram of Matsuoka's neural oscillator*

rate of the neuron, respectively. Self-inhibitory input $v_i(t)$ represents an adaptation or fatigue property that ubiquitously exists in real neurons. Neuron $i$ receives an excitatory tonic input $c(> 0)$, which induces a spontaneous firing of each neuron, and an inhibitory input $-ay_j(t)$ from the other neuron $j(= i)$. Although function $g(x)$ is nonlinear in the general terminology, it has a linearity in a limited sense: $g(kx) = kg(x), (ke0)$. This scaling linearity will make analytical treatment of the present oscillator considerably easier.

**An example of the oscillation generated by the oscillator** An example of the oscillations generated by the oscillator is shown in Fig. 2.9. Figure 2.9a shows a time course of variables $x_1(t)$ (a thin line) and $y_1(t)$ (a thick line); their counterparts, $x_2(t)$ and $y_2(t)$, are just antiphasic to $x_1(t)$ and $y_1(t)$. Because of the symmetric structure of the oscillator, it is sometimes convenient to transform the state variables as $x(t) = x_1(t) - x2(t)$, $v(t) = {}_1(t) - {}_2(t)$, $X(t) = x_1(t) + x_2(t)$, and $V(t) = {}_1(t) + {}_2(t)$. The time courses of these variables are shown in Fig. 2.9b and 2.9c. In this example, the model parameters are set as follows: a = 2.5, b = 2.5, c = 1.5, $\tau$ = 0.25, and T = 0.5. This setting of the parameters will be used as a reference one in all the simulations shown later, where only parameter $T$ or $a$ will be varied.

**2.3.1.2.2 Rowat-Selverston Model** Biological neurons with several ionic channels are complex, hence difficult to model. Rowat and Selverston [163, 162, 161] present a simple model of a neuron for which two groups of currents are identified: a fast current and a slow

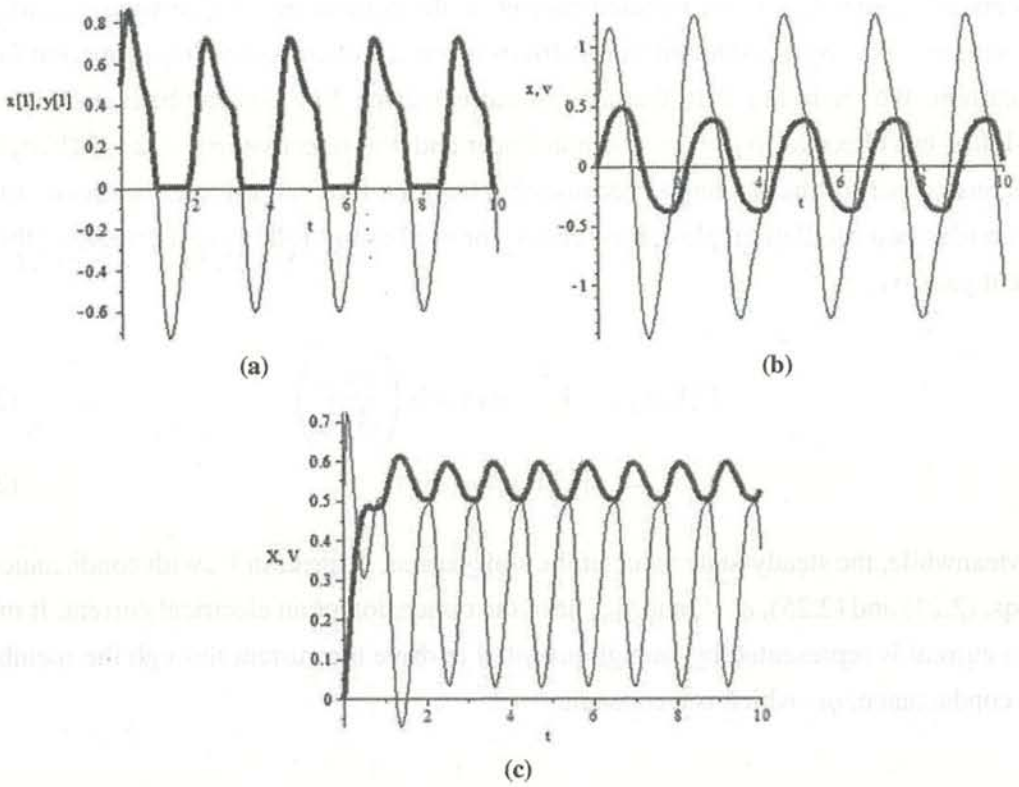



**Figure 2.9:** *An oscillation generated by the neural oscillator. a) $x_1(t)$ (thin line) and $y_1(t)$ (thick line), in which $x_1(t) = y_1(t)$ for $y_1(t) > 0$; b) $x(t)$ (thin line) and $(t)$ (thick line); c) $X(t)$ (thin line) and $V(t)$ (thick line)*

current, each defined by a first order differential equation. Fast current is defined by Eq. (2.24) and slow current by Eq. (2.25).

$$\tau_m \frac{dV}{dt} = -\left( F(V, \sigma_f) + q - I_{inj} \right) \tag{2.24}$$

$$\tau_s \frac{dq}{dt} = -q + q_\infty(V) \tag{2.25}$$

with $\tau_m < \tau_s$. $\tau_m$, is the time constant of the neuron membrane, $\tau_s$ is the time constant of slow currents activation. $I_{inj}$, is the injected current, $V$ the cellular membrane voltage, and $q$ the slow current. $F(V, \sigma_f)$, defined in Eq. (2.26), is a non-linear current-voltage function for the fast current. We see in Fig. 2.10 that the fast current shape $F(V, \sigma_f)$ can be linear ($\sigma_f = 0$), non-linear but bijective ($\sigma_f = 1$), and non-linear and non-bijective ($\sigma_f = 2$). $F(V, \sigma_f)$ is a fundamental part of the RS model because this function induces different behaviors for the neuron (damped oscillating, plateau potentials, or oscillating) following the value of the fast current gain, $\sigma_f$.

$$F(V, \sigma_f) = V - A_f \tanh\left( \frac{\sigma_f}{A_f} V \right) \tag{2.26}$$

$$q_\infty(V) = \sigma_s V \tag{2.27}$$

Meanwhile, the steady-state value of the slow current is linear in $V$, with conductance $\sigma_s$. In Eqs. (2.24) and (2.25), $q$, $V$, and $I_{inj}$ have the dimension of an electrical current. It means that a current is represented by enough potential to drive the current through the membrane leak conductance, $g_L$, which is a constant.

#### 2.3.1.3 Passive Locomotion [61]

Passive dynamic is the novel control method based on point-contact and virtual holonomic constraint [45]. Point-contact denotes that a robot contacts the ground at a point (i.e. the first joint is passive), and makes it possible to achieve adaptability to ground irregularity and energy efficiency. The concept of the virtual holonomic constraint is proposed as Virtual constraint by Grizzle and Westervelt et al. [65, 214], and defined as a set of holonomic constraints on the robot's actuated DoF parameterized by the robot's unactuated DoF. The virtual holonomic constraint enables a robot to satisfy the desired path of postural motion.

### 2.3.2 Stability Model

In order to support the stability of the legged robot, stability model is required to be installed. It will keep the stability of the robot from internal or external disturbance. In current situation, there are several models proposed by researchers.

#### 2.3.2.1 Zero Moment Point

Zero moment point is a concept related with dynamics and control of legged locomotion, e.g., for humanoid robots. It specifies the point with respect to which dynamic reaction force at the contact of the foot with the ground does not produce any moment in the horizontal direction, i.e. the point where the total of horizontal inertia and gravity forces equals 0 (zero). The concept assumes the contact area is planar and has sufficiently high friction to keep the feet from sliding.

The zero moment point is a very important concept in the motion planning for biped robots. Since they have only two points of contact with the floor and they are supposed to walk, "run" or "jump" (in the motion context), their motion has to be planned concerning the dynamical stability of their whole body. This is not an easy task, especially because the upper body of the robot (torso) has larger mass and inertia than the legs which are supposed to support and move the robot. This can be compared to the problem of balancing an inverted pendulum.

The trajectory of a walking robot is planned using the angular momentum equation to ensure that the generated joint trajectories guarantee the dynamical postural stability of the robot, which usually is quantified by the distance of the zero moment point in the boundaries of a predefined stability region. The position of the zero moment point is affected by the referred mass and inertia of the robot's torso, since its motion generally requires large ankle torques to maintain a satisfactory dynamical postural stability.

One approach to solve this problem consists in using small trunk motions to stabilize the posture of the robot. However, some new planning methods are being developed to define the trajectories of the legs' links in such a way that the torso of the robot is naturally steered in order to reduce the ankle torque needed to compensate its motion. If the trajectory planning for the leg links is well succeeded, then the zero moment point won't move out of the predefined stability region and the motion of the robot will become smoother, mimicking a natural trajectory.

The resultant force of the inertia and gravity forces acting on a biped robot is expressed by the formula:

$$F^{gi} = mg - ma_G \tag{2.28}$$

where $m$ is the total mass of the robot, $g$ is the acceleration of the gravity, $G$ is the center of mass and $a_G$ is the acceleration of the center of mass. The moment in any point $X$ can be defined as:

$$M_X^{gi} = \overrightarrow{XG} \times mg - \overrightarrow{XG} \times ma_G - \dot{H}_G \tag{2.29}$$

where $\dot{H}_G$ is the rate of angular momentum at the center of mass.

The Newton–Euler equations of the global motion of the biped robot can be written as:

$$F^c + mg = ma_G \tag{2.30}$$

$$M_X^c + \overrightarrow{XG} \times mg = \dot{H}_G + \overrightarrow{XG}. \times ma_G \tag{2.31}$$

where $F^c$ is the resultant of the contact forces at X and $M_X^c$ is the moment related with contact forces about any point X. The Newton–Euler equations can be rewritten as:

$$F^c + (mg - ma_G) = 0 \tag{2.32}$$

$$M_X^c + (\overrightarrow{XG} \times mg - \overrightarrow{XG} \times ma_G - \dot{H}_G) = 0 \tag{2.33}$$

so it's easier to see that we have:

$$F^c + F^{gi} = 0 \tag{2.34}$$

$$M_X^c + M_X^{gi} = 0 \tag{2.35}$$

These equations show that the biped robot is dynamically balanced if the contact forces and the inertia and gravity forces are strictly opposite. If an axis $\Delta^{gi}$ is defined, where the moment is parallel to the normal vector $n$ from the surface about every point of the axis, then the Zero Moment Point (ZMP) necessarily belongs to this axis, since it is by definition directed along the vector $n$. The ZMP will then be the intersection between the axis

$\Delta^{gi}$ and the ground surface such that:

$$M_Z^{gi} = \overrightarrow{ZG} \times mg - \overrightarrow{ZG} \times ma_G - \dot{H}_G \tag{2.36}$$

with

$$M_Z^{gi} \times n = 0 \tag{2.37}$$

where $Z$ represents the ZMP. Because of the opposition between the gravity and inertia forces and the contact forces mentioned before, the $Z$ point (ZMP) can be defined by:

$$\overrightarrow{PZ} = \frac{n \times M_P^{gi}}{F^{gi} \cdot n} \tag{2.38}$$

where $P$ is a point on the contact plane, e.g. the normal projection of the center of mass.

#### 2.3.2.2 Center of Gravity

The center of gravity is a geometric property of any object. The center of gravity is the average location of the weight of an object. We can completely describe the motion of any object through space in terms of the translation of the center of gravity of the object from one place to another, and the rotation of the object about its center of gravity if it is free to rotate. If the object is confined to rotate about some other point, like a hinge, we can still describe its motion. In flight, both airplanes and rockets rotate about their centers of gravity. A kite, on the other hand, rotates about the bridle point. But the trim of a kite still depends on the location of the center of gravity relative to the bridle point, because for every object the weight always acts through the center of gravity.

Determining the center of gravity is very important for any flying object. In general, determining the center of gravity (CoG) is a complicated procedure because the mass (and weight) may not be uniformly distributed throughout the object. The general case requires the use of calculus which we will discuss at the bottom of this page. If the mass is uniformly distributed, the problem is greatly simplified. If the object has a line (or plane) of symmetry, the cg lies on the line of symmetry. For a solid block of uniform material, the center of gravity is simply at the average location of the physical dimensions. (For a rectangular block, 50 X 20 X 10, the center of gravity is at the point (25, 10, 5) ). For a triangle of height h, the cg is at h/3, and for a semi-circle of radius r, the cg is at $(4 * r/(3 * \pi))$ where pi is ratio of the circumference of the circle to the diameter. There are tables of the location of the center of gravity for many simple shapes in math and science books. The tables were generated by using the equation from calculus shown on the slide.

For a general shaped object, there is a simple mechanical way to determine the center of gravity:

- If we just balance the object using a string or an edge, the point at which the object is balanced is the center of gravity. (Just like balancing a pencil on your finger!)

- Another, more complicated way, is a two step method shown on the slide. In Step 1, you hang the object from any point and you drop a weighted string from the same point. Draw a line on the object along the string. For Step 2, repeat the procedure from another point on the object You now have two lines drawn on the object which intersect. The center of gravity is the point where the lines intersect. This procedure works well for irregularly shaped objects that are hard to balance. If the mass of the

object is not uniformly distributed, we must use calculus to determine center of gravity. We will use the symbol S dw to denote the integration of a continuous function with respect to weight. Then the center of gravity can be determined from:

$$cg \times W = Sxdw \tag{2.39}$$

where x is the distance from a reference line, dw is an increment of weight, and W is the total weight of the object. To evaluate the right side, we have to determine how the weight varies geometrically. From the weight equation, we know that:

$$w = m \times g \tag{2.40}$$

where m is the mass of the object, and g is the gravitational constant. In turn, the mass m of any object is equal to the density, rho, of the object times the volume, V:

$$m = \rho \times V \tag{2.41}$$

We can combine the last two equations:

$$w = g \times \rho \times V \tag{2.42}$$

then

$$dw = g \times \rho \times dV \tag{2.43}$$

$$dw = g \times \rho(x, y, z) \times dxdydz \tag{2.44}$$

If we have a functional form for the mass distribution, we can solve the equation for the center of gravity:

$$cg \times W = g \times SSSx \times \rho(x, y, z)dxdydz \tag{2.45}$$

where $SSS$ indicates a triple integral over $dx$. $dy$. and $dz$. If we don't know the functional form of the mass distribution, we can numerically integrate the equation using a spreadsheet. Divide the distance into a number of small volume segments and determining the average value of the weight/volume (density times gravity) over that small segment. Taking the sum of the average value of the weight/volume times the distance times the volume segment divided by the weight will produce the center of gravity.

### 2.3.3 Motion Planning Model [1]

Motion planning (also known as the navigation problem or the piano mover's problem) is a term used in robotics for the process of breaking down a desired movement task into discrete motions that satisfy movement constraints and possibly optimize some aspect of the movement.

For example, consider navigating a mobile robot inside a building to a distant waypoint. It should execute this task while avoiding walls and not falling down stairs. A motion planning algorithm would take a description of these tasks as input, and produce the speed and turning commands sent to the robot's wheels. Motion planning algorithms might address robots with a larger number of joints (e.g., industrial manipulators), more complex tasks (e.g. manipulation of objects), different constraints (e.g., a car that can only drive forward), and uncertainty (e.g. imperfect models of the environment or robot).

Motion planning has several robotics applications, such as autonomy, automation, and robot design in CAD software, as well as applications in other fields, such as animating digital characters, video game artificial intelligence, architectural design, robotic surgery, and the study of biological molecules.

#### 2.3.3.1 Concept

A basic motion planning problem is to produce a continuous motion that connects a start configuration S and a goal configuration G, while avoiding collision with known obstacles. The robot and obstacle geometry is described in a 2D or 3D workspace, while the motion is represented as a path in (possibly higher-dimensional) configuration space.

**2.3.3.1.1 Configuration space** A configuration describes the pose of the robot, and the configuration space C is the set of all possible configurations. For example:

- If the robot is a single point (zero-sized) translating in a 2-dimensional plane (the workspace), C is a plane, and a configuration can be represented using two parameters (x, y).

- If the robot is a 2D shape that can translate and rotate, the workspace is still 2-dimensional. However, C is the special Euclidean group $\mathbf{SE}(2) = \mathbf{R}^2 \times \mathbf{SO}(2)$ (where $\mathbf{SO}(2)$ is the special orthogonal group of 2D rotations), and a configuration can be represented using 3 parameters (x, y, $\theta$ ).

- If the robot is a solid 3D shape that can translate and rotate, the workspace is 3-dimensional, but C is the special Euclidean group $\mathbf{SE}(3) = \mathbf{R}^3 \times SO(3)$, and a configuration requires 6 parameters: (x, y, z) for translation, and Euler angles $(\alpha, \beta, \gamma)$.
- If the robot is a fixed-base manipulator with N revolute joints (and no closed-loops), C is N-dimensional.

**2.3.3.1.2 Free space** The set of configurations that avoids collision with obstacles is called the free space Cfree. The complement of Cfree in C is called the obstacle or forbidden region.

Often, it is prohibitively difficult to explicitly compute the shape of Cfree. However, testing whether a given configuration is in Cfree is efficient. First, forward kinematics determine the position of the robot's geometry, and collision detection tests if the robot's geometry collides with the environment's geometry.

**2.3.3.1.3 Target space** Target space is a linear subspace of free space which denotes where we want the robot to move to. In global motion planning, target space is observable by the robot's sensors. However, in local motion planning, the robot cannot observe the target space in some states. To solve this problem, the robot goes through several virtual target spaces, each of which is located within the observable area (around the robot). A virtual target space is called a sub-goal. [109]

### 2.3.3.2 Algorithm

Low-dimensional problems can be solved with grid-based algorithms that overlay a grid on top of configuration space, or geometric algorithms that compute the shape and connectivity of $C_{free}$.

Exact motion planning for high-dimensional systems under complex constraints is computationally intractable. Potential-field algorithms are efficient, but fall prey to local minima (an exception is the harmonic potential fields). Sampling-based algorithms avoid the problem of local minima, and solve many problems quite quickly. They are unable to determine that no path exists, but they have a probability of failure that decreases to zero as more time is spent.

Sampling-based algorithms are currently considered state-of-the-art for motion planning in high-dimensional spaces, and have been applied to problems which have dozens or even hundreds of dimensions (robotic manipulators, biological molecules, animated digital characters, and legged robots).

**2.3.3.2.1 Grid-based Search** Grid-based approaches overlay a grid on configuration space, and assume each configuration is identified with a grid point. At each grid point, the robot is allowed to move to adjacent grid points as long as the line between them is completely contained within $C_{free}$ (this is tested with collision detection). This discretizes the set of actions, and search algorithms (like A*) are used to find a path from the start to the goal.

These approaches require setting a grid resolution. Search is faster with coarser grids, but the algorithm will fail to find paths through narrow portions of $C_{free}$. Furthermore, the number of points on the grid grows exponentially in the configuration space dimension, which make them inappropriate for high-dimensional problems.

Traditional grid-based approaches produce paths whose heading changes are constrained to multiples of a given base angle, often resulting in suboptimal paths. Any-angle path planning approaches find shorter paths by propagating information along grid edges (to search fast) without constraining their paths to grid edges (to find short paths).

Grid-based approaches often need to search repeatedly, for example, when the knowledge of the robot about the configuration space changes or the configuration space itself changes during path following. Incremental heuristic search algorithms replan fast by using experience with the previous similar path-planning problems to speed up their search for the current one.

**2.3.3.2.2 Interval-based search** These approaches are similar to grid-based search approaches except that they generate a paving covering entirely the configuration space instead of a grid.[2] The paving is decomposed into two subpavings X−,X+ made with boxes such that X− ⊂ Cfree ⊂ X+. Characterizing Cfree amounts to solve a set inversion problem. Interval analysis could thus be used when Cfree cannot be described by linear inequalities in order to have a guaranteed enclosure.

The robot is thus allowed to move freely in X−, and cannot go outside X+. To both subpavings, a neighbor graph is built and paths can be found using algorithms such as Dijkstra or A*. When a path is feasible in X−, it is also feasible in Cfree. When no path exists in X+ from one initial configuration to the goal, we have the guarantee that no feasible path exists in Cfree. As for the grid-based approach, the interval approach is inappropriate for high-dimensional problems, due to the fact that the number of boxes to be generated grows exponentially with respect to the dimension of configuration space.

An illustration is provided by the three figures on the right where a hook with two degrees of freedom has to move from the left to the right, avoiding two horizontal small segments.

Motion from the initial configuration (blue) to the final configuration of the hook, avoiding the two obstacles (red segments). The left-bottom corner of the hook has to stay on the

horizontal line, which makes the hook two degrees of freedom.

Decomposition with boxes covering the configuration space: The subpaving X— is the union all red boxes and the subpaving X+ is the union of red and green boxes. The path corresponds to the motion represented above.

This figure corresponds to the same path as above but obtained with many fewer boxes.The algorithm avoids bisecting boxes in parts of the configuration space that do not influence the final result. The decomposition with subpavings using interval analysis also makes it possible to characterize the topology of Cfree such as counting its number of connected components. [38]

**2.3.3.2.3 Reward-based algorithms** Reward-based algorithms assume that the robot in each state (position and internal state, including direction) can choose between different actions (motion). However, the result of each action is not definite. In other words, outcomes (displacement) are partly random and partly under the control of the robot. The robot gets positive reward when it reaches the target and gets negative reward if it collides with an obstacle. These algorithms try to find a path which maximizes cumulative future rewards. The Markov decision process (MDP) is a popular mathematical framework that is used in many reward-based algorithms. The advantage of MDPs over other reward-based algorithms is that they generate the optimal path. The disadvantage of MDPs is that they limit the robot to choose from a finite set of actions. Therefore, the path is not smooth (similar to grid-based approaches). Fuzzy Markov decision processes (FDMPs) are an extension of MDPs which generate smooth paths using a fuzzy inference system.[109]

**2.3.3.2.4 Artificial potential fields** One approach is to treat the robot's configuration as a point (usually electron) in a potential field that combines attraction to the goal, and repulsion from obstacles. The resulting trajectory is output as the path. This approach has advantages in that the trajectory is produced with little computation. However, they can become trapped in local minima of the potential field, and fail to find a path. The Artificial potential fields can be achieved by direct equation similar to electrostatic potential fields or can be drive by set of linguistic rules

**2.3.3.2.5 Sampling-based algorithm** Sampling-based algorithms represent the configuration space with a roadmap of sampled configurations. A basic algorithm samples N configurations in C, and retains those in Cfree to use as milestones. A roadmap is then constructed that connects two milestones P and Q if the line segment PQ is completely in Cfree. Again, collision detection is used to test inclusion in Cfree. To find a path that connects S and G,

they are added to the roadmap. If a path in the roadmap links S and G, the planner succeeds, and returns that path. If not, the reason is not definitive: either there is no path in Cfree, or the planner did not sample enough milestones.

These algorithms work well for high-dimensional configuration spaces, because unlike combinatorial algorithms, their running time is not (explicitly) exponentially dependent on the dimension of C. They are also (generally) substantially easier to implement. They are probabilistically complete, meaning the probability that they will produce a solution approaches 1 as more time is spent. However, they cannot determine if no solution exists.

Given basic visibility conditions on Cfree, it has been proven that as the number of configurations N grows higher, the probability that the above algorithm finds a solution approaches 1 exponentially.[5] Visibility is not explicitly dependent on the dimension of C; it is possible to have a high-dimensional space with "good" visibility or a low-dimensional space with "poor" visibility. The experimental success of sample-based methods suggests that most commonly seen spaces have good visibility.

There are many variants of this basic scheme:

- It is typically much faster to only test segments between nearby pairs of milestones, rather than all pairs.
- Nonuniform sampling distributions attempt to place more milestones in areas that improve the connectivity of the roadmap.
- Quasirandom samples typically produce a better covering of configuration space than pseudorandom ones, though some recent work argues that the effect of the source of randomness is minimal compared to the effect of the sampling distribution.
- It is possible to substantially reduce the number of milestones needed to solve a given problem by allowing curved eye sights (for example by crawling on the obstacles that block the way between two milestones [20]).
- If only one or a few planning queries are needed, it is not always necessary to construct a roadmap of the entire space. Tree-growing variants are typically faster for this case (single-query planning). Roadmaps are still useful if many queries are to be made on the same space (multi-query planning)

#### 2.3.3.3 Completeness and performance

A motion planner is said to be complete if the planner in finite time either produces a solution or correctly reports that there is none. Most complete algorithms are geometry-

based. The performance of a complete planner is assessed by its computational complexity.

Resolution completeness is the property that the planner is guaranteed to find a path if the resolution of an underlying grid is fine enough. Most resolution complete planners are grid-based or interval-based. The computational complexity of resolution complete planners is dependent on the number of points in the underlying grid, which is $O(1/h^d)$, where h is the resolution (the length of one side of a grid cell) and d is the configuration space dimension.

Probabilistic completeness is the property that as more "work" is performed, the probability that the planner fails to find a path, if one exists, asymptotically approaches zero. Several sample-based methods are probabilistically complete. The performance of a probabilistically complete planner is measured by the rate of convergence. *Incomplete* planners do not always produce a feasible path when one exists. Sometimes incomplete planners do work well in practice.

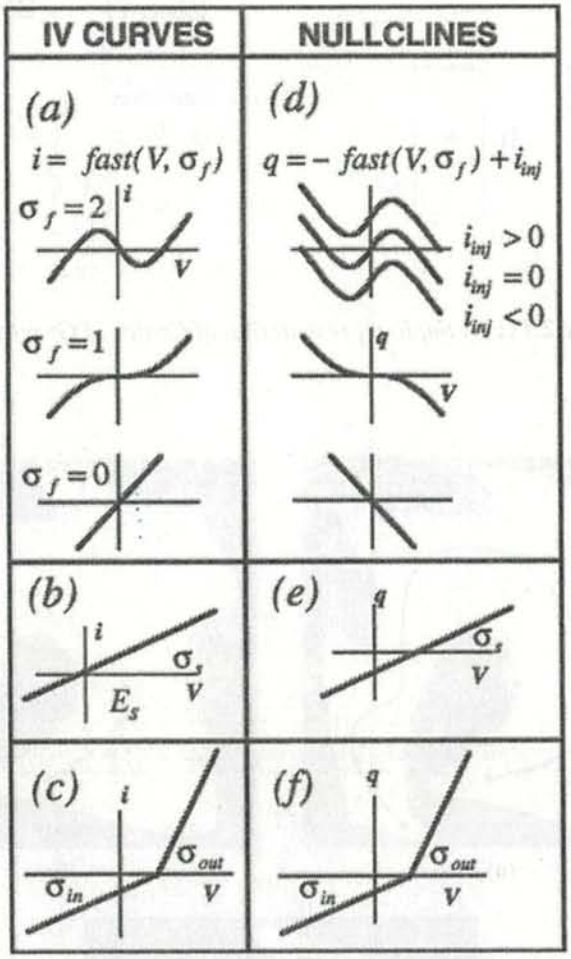



**Figure 2.10:** *IV curves and nullclines in the cell model. (a) The curve $i = F(V, \sigma_f)$ for three values of $\sigma_f$. (b) The standard slow current IV curve. ES is the reversal potential. (c) The split slow current IV curve with different inward and outward conductances. (d) The V-nullcline for $\sigma_f = 2, 1, 0$. For $\sigma_f = 2$, three values of the injected current produce three positions of the $V$-nullcline. (e) The q-nullcline. (f) The q-nullcline when using a split slow current. A fast current null cline is obtained by reflecting the IV curve of the fast current in the $V$-axis and then moving it up or down by the amount of the injected current. A slow current nullcline is identical with the IV curve of the slow current. [161]*

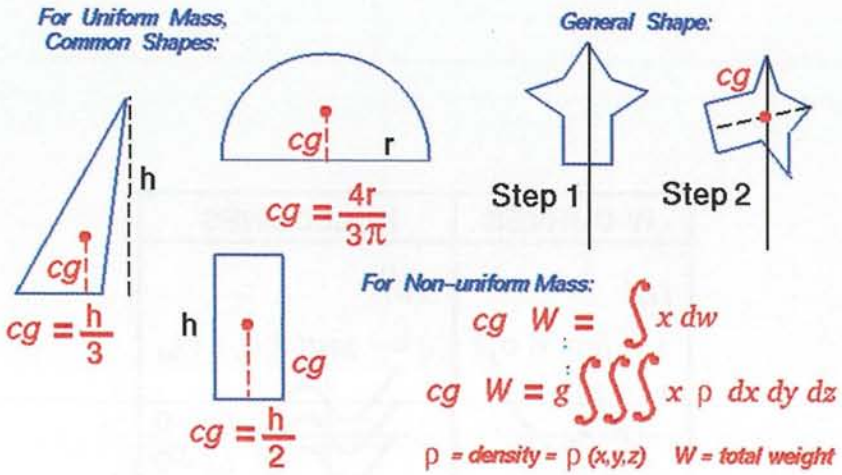



**Figure 2.11:** *Example representation of Center of Gravity model*

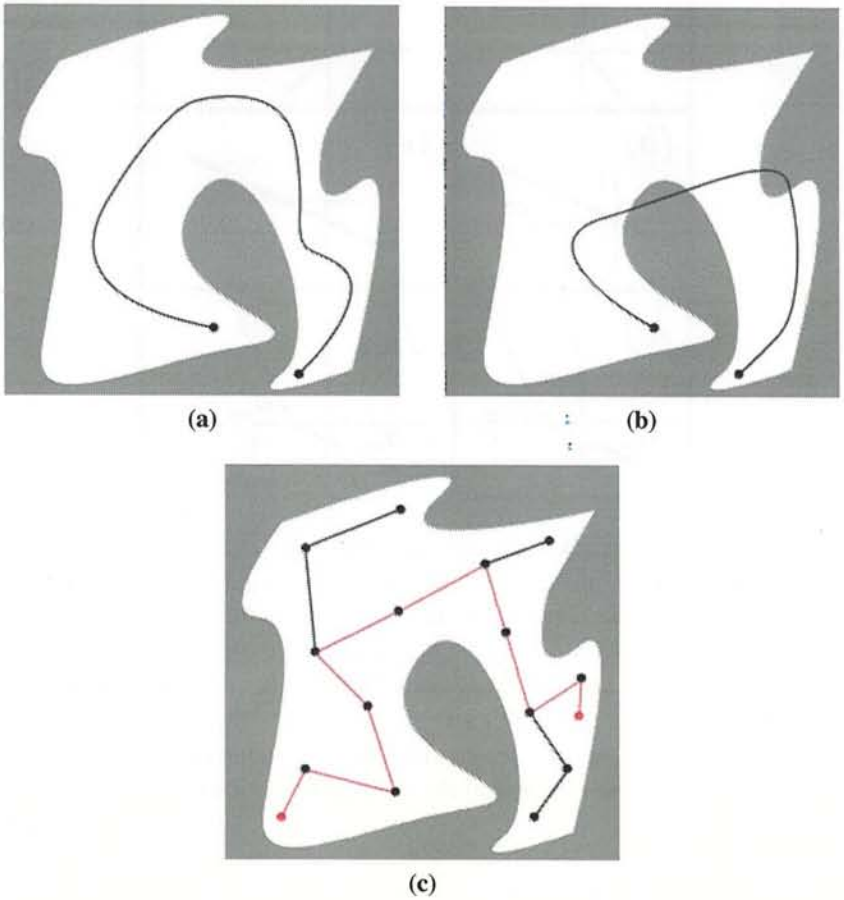



**Figure 2.12:** *(a) Example of a valid path (b) example of invalid path (c) example of a road map*

# Chapter 3

# Computational Intelligence

In developing motion capabilities in the robot systems, computational intelligence (CI) is the technology to realize adaptive motion planning behavior in the legged robot. In this chapter, we will discuss about CI used in this thesis including fuzzy computing, neuro computing, and evolutionary computing for intelligence of robotics.

## 3.1 Fuzzy Computing

Human reasoning and other natural phenomena cannot be accurately described by only two-valued logic. Therefore, there is an idea to extend two-valued logic into multiple valued ones. In other word, instead of using only the "true" and "false" logical values, the use of other values between them also should be allowed. Fuzzy logic expresses the values between "true" and "false" using degree of truth represented as a value in membership function. Membership function sample of human height can be seen in Fig. 3.1.

Conducting fuzzy computing requires several steps, such as fuzzification, inference mechanism, and defuzzification. These steps can be shown in Fig. 3.2. Fuzzification is performed by calculating crisp value into membership function based on degree of truth that was defined before hand. The fuzzy rule is defined by experts. The general form of a fuzzy rule with one input and one output dimension can be defined as follows :

$R$ :**If** $x$ is $A$ **then** $y$ is $B$,

where $x \in X$ is the input and $y \in Y$ is the output variable, $X$ is the universe of discourse for the input and $Y$ is the universe of discourse for the output variable. $A$ and $B$ are linguistic labels which are expressed by fuzzy sets. Set $A$ is the antecedent while $B$ is the consequent of rule $R$. The extended of the above rule which is the general form of a fuzzy rule with multiple inputs and one output dimension can be written in the following, which is called the

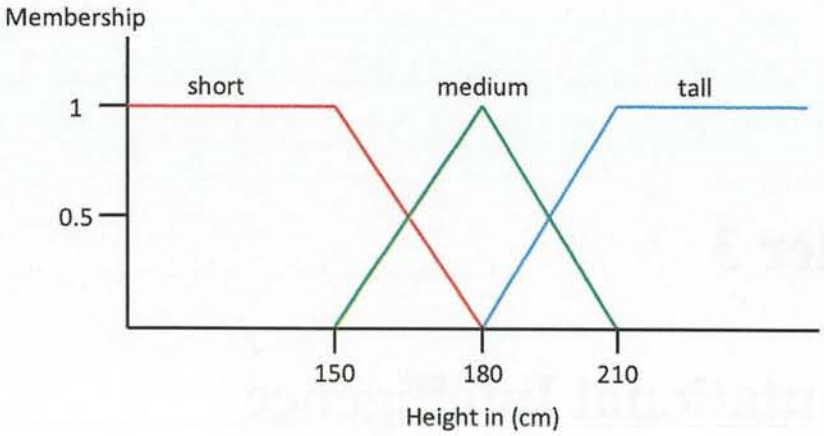



**Figure 3.1:** *Membership function*

Mamdani type orthogonally decomposed form [123]:

$R$ :**If** $x_1$ is $A_1$ **and ... and** $x_n$ is $A_n$ **then** $y$ is $B$,

where $x = (x_1, ..., x_n)$ is the input vector, $x_j \in X_j$, $X = X_1 \times ... \times X_n$ is the $n-$dimensional universe, $A = (A_1, ..., A_n)$ is the antecedent vector, $A \subset X$, $y \in Y$ is the output variable, $Y$ is the universe for the output and $B$ is the consequent set, $B \subset Y$ (Here $\subset$ denotes fuzzy subsethood.). A rule can be applied if every input variable has a positive membership value in its corresponding antecedent set. In the case of multiple output rules, the outputs are independent from each other, thus this kind of rules can be decomposed to fuzzy rules with one single output, which can reduce the computational calculation.

After defining fuzzy rules, the next step is conducting inference system. In fuzzy computing there are some well-known inference technique, the Mamdani method is the most commonly used in practical application [123]. The Mamdani inference algorithm can be illustrated with fuzzy membership functions in Fig. 3.3.

At the beginning of the inference the degree of matching between the observation and the rules is determined. Each component of the observation vector is compared to the same component of the antecedent of each rule. Let $A^*$ be the $n-$dimensional observation vector. The degree of matching (firing) in the$j$-th dimension in the $i$-th rule can be computed as:

$$w_{j,i} = \max_{x_j}\{\min\{A_j^*(x_j), A_{j,i}(x_j)\}\} \tag{3.1}$$

where $A_{j,i}$ is the membership function of the $i$-th rule in the $j$-th dimension. If the observation is a crisp vector then the above calculation is simpler: in case of state vector $x^*$, the

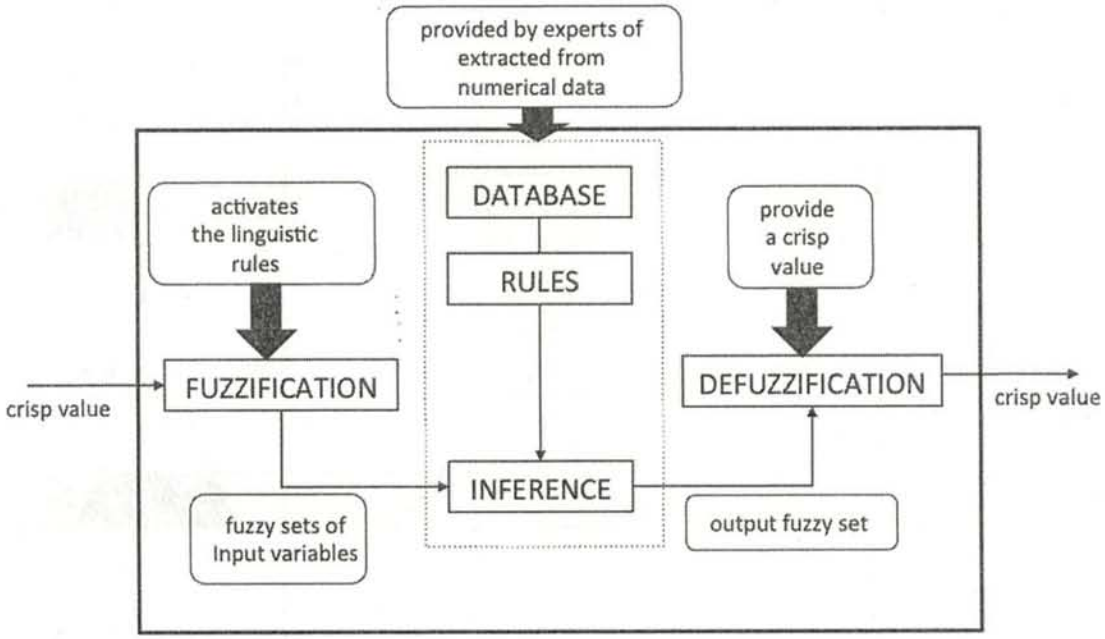



**Figure 3.2:** *Fuzzy computing*

degree of matching in the $j$-th dimension is:

$$w_{j,i} = A_{j,i}(x_j^*) \tag{3.2}$$

After the degree of matching was calculated in each dimension, the resultant for the whole antecedent is determined. The degree of applicability of a rule is affected by the degree of matching of its each dimension. Thus, the firing degree of the $i$-th rule can be computed by taking the minimum value of the degrees of matching of the rule's antecedents:

$$w_i = \min_{j=1}^{n} w_{j,i} \tag{3.3}$$

$w_i$ shows that how important the role of rule $R_i$ will be in the calculation of the conclusion for observation $A^*$.

After the degree of firing was determined for each rule, each conclusion is separately calculated. This can be made by cutting the consequent fuzzy set of the rule at height $w_i$:

$$B_i^* = \min(w_i, B_i(y)) \tag{3.4}$$

The conclusion for the whole rule base can be computed by taking the union of the

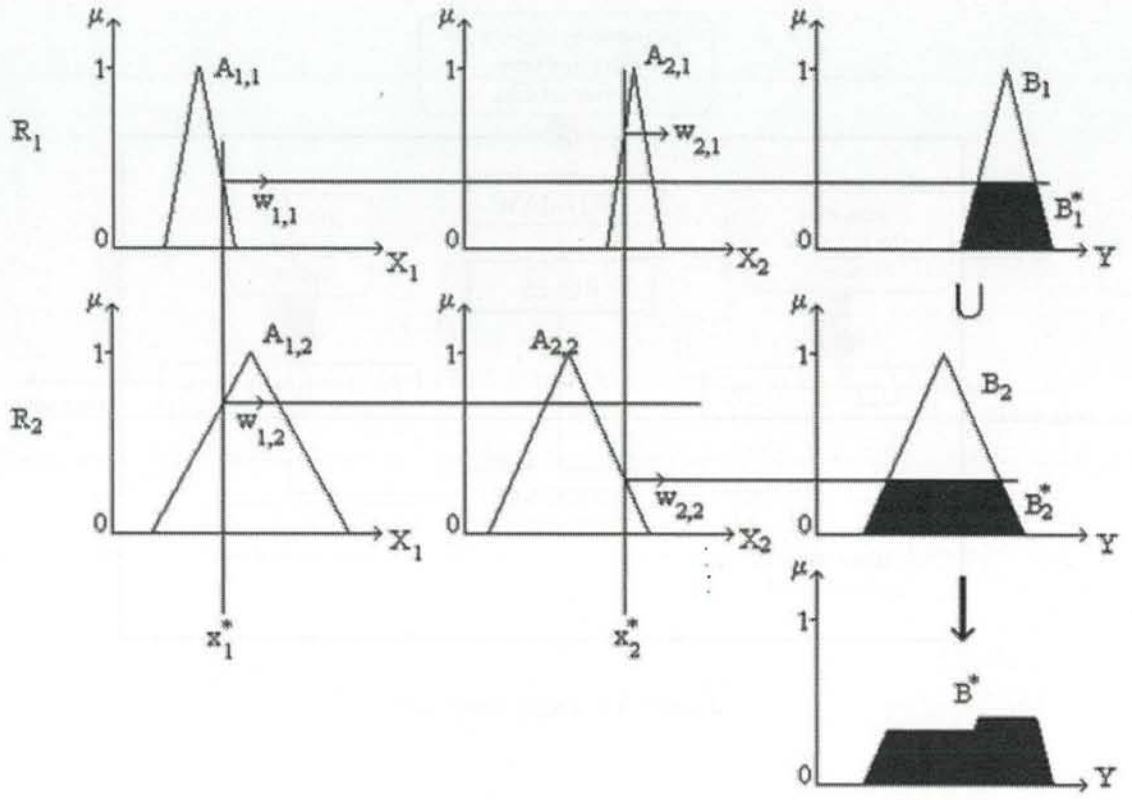



**Figure 3.3:** *Mamdani inference algorithm*

previously calculated sub-conclusions:

$$B^* = \max_{i=i}^{r} B_i^*(y) \tag{3.5}$$

After the inference a $B^*(y)$ conclusion fuzzy set was obtained. However, in most of the cases, the expected conclusion is not a fuzzy set, but a crisp value. Hence, the crisp value needs to be determined, which describes the conclusion fuzzy set in the best way. This procedure is called defuzzification. There are many different defuzzification methods described in the literature, in this particular application the Centre of Gravity (COG) method is applied, which is one of the most commonly used defuzzification techniques in practical applications. The COG method provides a crisp result that can be calculated as follows:

$$y_{COG} = \frac{\sum_{t=1}^{r} \int_{y \in B_i^*} B_i^*(y) y dy}{\sum_{t=1}^{r} \int_{y \in B_i^*} B_i^*(y) dy} \tag{3.6}$$

## 3.2 Neuro Computing

### 3.2.1 Artificial neuron model

The long course of evolution has given the human brain many desirable characteristics not present in von Neumann or modern parallel computers. These include

- Massive parallelism
- Distributed representation and computation
- Learning ability
- Generalization ability
- Adaptivity
- Inherent contextual information processing
- Fault tolerance
- Low energy consumption

It is hoped that devices based on biological neural networks will possess some of these desirable characteristics.

Recently, modern digital computers has outperformed humans in numerical computation and related symbol manipulation. However, humans can easily solve complex perceptual problems such as recognizing a man in a crowd from a glimpse of his face in high speed comparing to even the world's fastest computer. Why is there such a remarkable difference in their performance? The reason behind this is biological neural system architecture is completely different from the modern parallel computers or von Neumann architecture as shown in Table 3.1. This difference affects the type of functions each computational model can perform.

In order to develop general-purpose intelligent programs, biological inspired neural networks or artificial neural networks (ANNs) is implemented. ANNs are parallel computing systems consist of a large number of simple processors with many interconnections. ANN models try to use some "organizational" principles used in the human brain.

As we know the human nervous consists of small unicellular structures called neurons. There are 100 billion neurons in the human brain. Neurons are the fundamental units or building blocks of a biological nervous system. Figure 3.4 show that neuron consisted of basic elements such as dendrite, soma (cell body), synapse and axon.

**Table 3.1:** *Modern parallel computer versus biological neural system*

| | Modern parallel computer | Biological neural system |
|---|---|---|
| Processor | Complex<br>High speed<br>One or a few | Simple<br>Low speed<br>A large number |
| Memory | Separate from a processor<br>Noncontent addressable | Integrated into processor<br>Distributed content addressable |
| Computing | Centralized<br>Sequential<br>Stored programs | Distributed<br>Parallel<br>Self-learning |
| Reliability | Very vulnerable | Robust |
| Expertise | Numerical and symbolic manipulations | Perceptual problems |
| Operating environment | Well-defined<br>Well-constrained | Poorly defined<br>Unconstraint |

Dendrites receive signals from other neurons, muscles or sensory organs and carry the signals to the soma (cell body). On the other hand soma generates a composite signal based on the strength of the signals received from the dendrites. The composite signal is transmitted to the synapse through axon.

Synapse is similar to a potential barrier which controls the flow of signal from the axon of one neuron to the dendrites of other neurons. The transmission of signals from one neuron to another at a synapse is a complex chemical process. Basically, a specific chemical called neuro-transmitter is released from the transmitting end of the synapse. When the potential barrier is low, the signal from the transmitting end can easily reach the other end of the synapse. On the contrary, only a small component of the signal can reach the other end of the synapse. Synapse can also control the flow of signal from one neuron to the others. When the influence of the synapse tends to activate the post-synaptic neuron, the synapse is called excitatory. On the other hand, when the synapse prohibits the passage of signal flow to the post-synaptic neuron, the synapse is called inhibitory.

Based on the biological characteristics of neuron McCulloch and Pitts proposed a basic model of neuron with a binary threshold unit as a computational model as depicted in Fig. 3.5. Here we can see that in this model a neuron is composed of input, output, synaptic strength, and activation function. The output of neuron is produced based on activation function after the summing of inputs. The output of neuron before and after activation function

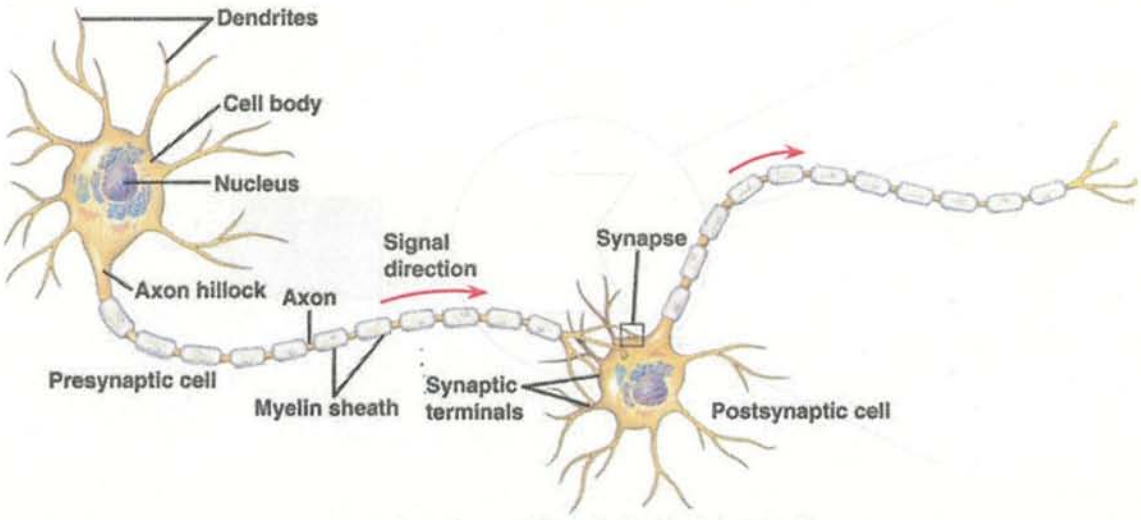



**Figure 3.4:** *Biological neuron*

can be calculated as follows.

$$h = \sum_{j=1}^{n} w_j x_j - \theta \tag{3.7}$$

$$y = f(h) = f\left(\sum_{j=1}^{n} w_j x_j - \theta\right) \tag{3.8}$$

where $x$ and $y$ are input and output respectively. $w$ is the weight parameter, $\theta$ id the threshold, and $n$ is the number of neurons. $f$ depicts the activation function, where basically step, sign, logistic, and hyperbolic tangent functions as shown in Fig. 3.6 are often used.

### 3.2.2 Network architectures

According to the connection pattern (architecture), ANNs can be divided into two categories as follows,

- Feed-forward networks (no loops occur)
- Recurrent (or feedback) networks (loops occur)

In feed-forward networks, the most common networks is multilayer perceptron. This networks are consisted of layers that have unidirectional connections between neurons. For other category of this group can be found in Fig. 3.7.

Feed-forward networks are static, which means that they produce only one set of output values rather than a sequence of values from a given input. This make feed- forward networks

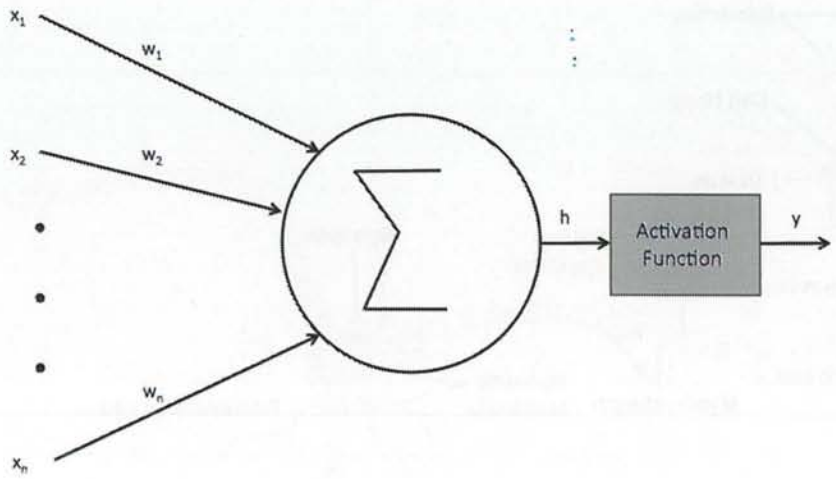



**Figure 3.5:** *McCulloh-Pitts model of neuron*

are memory-less in the sense that their response to an input is independent from the previous network state. However, recurrent networks are dynamic systems. When there is a new input pattern, the neuron outputs are computed. Because of the feedback characteristic, the inputs to each neuron are then modified, that leads the network to enter a new state.

### 3.2.3 Single-layer perceptron

Perceptron was proposed by Rosenblatt. According to the name, single layer perceptron only consist of single neuron with adjustable weight acquired from the learning algorithm. Perceptron learning algorithm can be depicted in algorithm 3.1.

**Algorithm 3.1** PERCEPTRON ALGORITHM

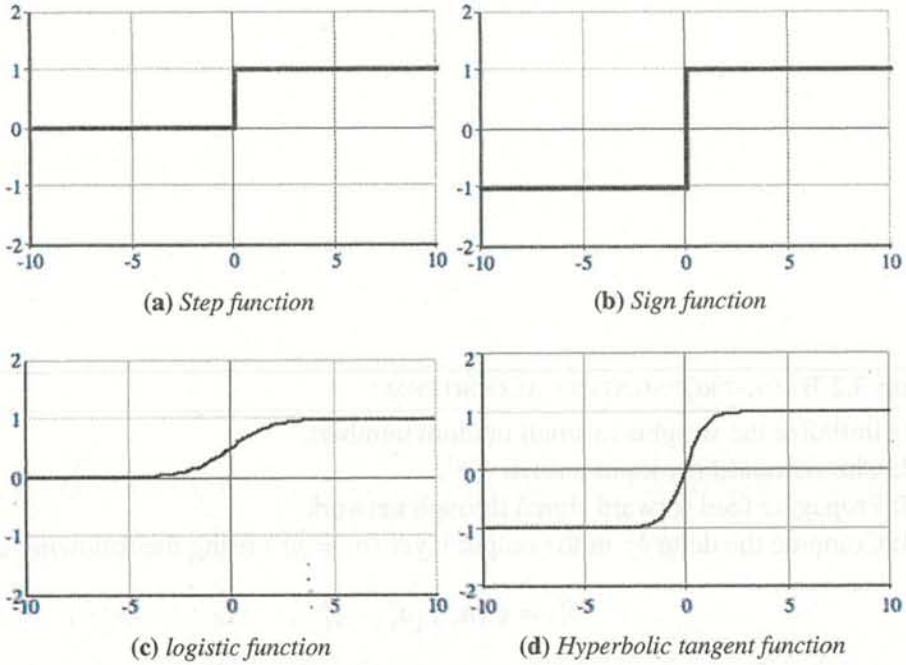
**Step 1:** Initialize the weights and threshold to small random numbers.
**Step 2:** Generate a pattern vector of $(x_1, x_2, ..., x_n)^t$ and evaluate the output of the neuron.
**Step 3:** Update the weights based on :

$$w_j(t+1) = w_j(t) + \eta(d-y)x_j, \tag{3.9}$$

where $d$ is the desired output, $t$ is the iteration number, and $\eta$ is the learning rate between (0,1)

### 3.2.4 Multilayer perceptron

Multilayer perceptron is used when the problem become complex or nonlinier problem. Different with single-layer perceptron, the multilayer perceptron is composed of many neu-




(a) *Step function* (b) *Sign function*

(c) *logistic function* (d) *Hyperbolic tangent function*

**Figure 3.6:** *Activation functions*

ron in hierarchical structure with one or more hidden layer as shown in Fig 3.8. The number of hidden layer is depend on the complexity of the problem itself. Figure 3.9 shows complex decision boundary from single-layer perceptron until three layer perceptron. In multilayer perceptron sigmoid activation function can be used to make a smooth decision boundary.

In multilayer perceptron back-propagation is among the most popular learning algorithm. The back-propagation algorithm is a gradient-descent method for minimizing the squared-error cost function.

The back-propagation algorithm can be written as shown in algorithm 3.2

### 3.2.5 Spiking Neurons

Generally, ANN can be divided into pulse-coded and rate-coded neuron models from the view point of abstraction [63]. The pulse-coded neuron approximates the dynamics created from the ignition phenomenon of neuron and also simulates propagation mechanism of the pulses between neuron. The pulse-coded neuron also known as spiking neuron. One of the classical neuronal spiking model is Hodgkin-Huxley model. This model uses four differential equations. On the other hand, the rate-coded neuron neglects the pulse structure and is considered to have higher level of abstraction. The McCulloch-Pitts model of ANN is well known as the rate-code model, while perceptron was proposed as a rate coded neural network

**Algorithm 3.2** BACK-PROPAGATION ALGORITHM

**Step 1:** Initialize the weights to small random numbers.
**Step 2:** Choose randomly input pattern $\mathbf{x}^{(\mu)}$.
**Step 3:** Propagate feed forward signal through network
**Step 4:** Compute the delta $\delta_i^L$ in the output layer $(o_i = y_i^L)$ using the following equation

$$\delta_i^L = g'(h_i^L)\left[d_i^u - y_i^L\right], \tag{3.10}$$

where $h_i^l$ is the net input to the $i$-th unit in the $L$-layer, and $g'$ is the derivative of the activation function $g$.
**Step 5:** Compute the deltas for the preceding layers by propagating the errors backwards using the following equiation

$$\delta_i^l = g'(h_i^l)\sum_j w_{ij}^{l+1}\delta_j^{l+1}, \tag{3.11}$$

for $l = (L-1), ..., 1$.
**Step 6:** Update the weights using the following equation

$$\Delta w_{ji}^l = \eta \delta_i^l y_j^{l-1} \tag{3.12}$$

**Step 7:** Go to the step 2 and repeat for the next pattern until the error in the output layer below the prespecified threshold or maximum number of iteration is reached.

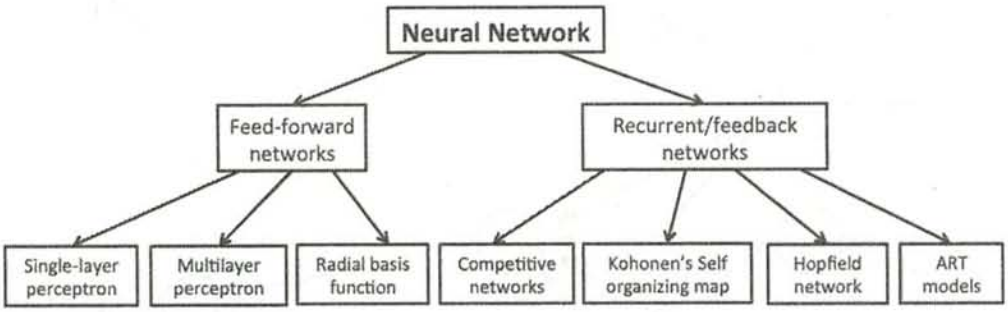



**Figure 3.7:** *Neural network architecture*

[15].

One of the important features of spiking neurons is the capability in conducting temporal coding. Spiking neural networks have been implemented in memorizing spatial and temporal context. Therefore, spiking neural networks are used to represent the time series of perceptual information. In this research the modified simple spike response model is used to reduce computational cost.

Basically, the calculation in spiking neural network is conducted through the following steps.

The membrane potential, or internal state $h_i(t)$ of the $i$-th neuron at the discrete time $t$ is given by:

$$h_i(t) = \tanh(h_i^{syn}(t) + h_i^{ref}(t) + h_i^{ext}(t)) \tag{3.13}$$

where $h_i^{syn}(t)$ includes the pulse outputs from the other neurons, $h_i^{ref}(t)$ is used for representing the refractoriness of the neuron, $h_i^{ext}(t)$ is the input to the $i$-th neuron from the environment. The hyperbolic tangent function is used to avoid the bursting of neuronal fires.

The first term $h_i^{syn}(t)$ is calculated as follows:

$$h_i^{syn}(t) = \gamma^{syn} \cdot h_i(t-1) + \sum_{j=1, j\neq i}^{N} w_{j,i} \cdot h_j^{PSP}(t-1) \tag{3.14}$$

where $\gamma^{syn}$ is the temporal discount rate, $w_{j,i}$ is a weight from the $j$-th neuron to the $i$-th neuron, $h_j^{PSP}(t)$ is the presynaptic action potential (PSP) approximately transmitted from the $j$-th neuron at the discrete time $t$, and $N$ is the number of neurons. When the internal state of the $i$-th neuron reaches the predefined threshold, a pulse is outputted as follows:

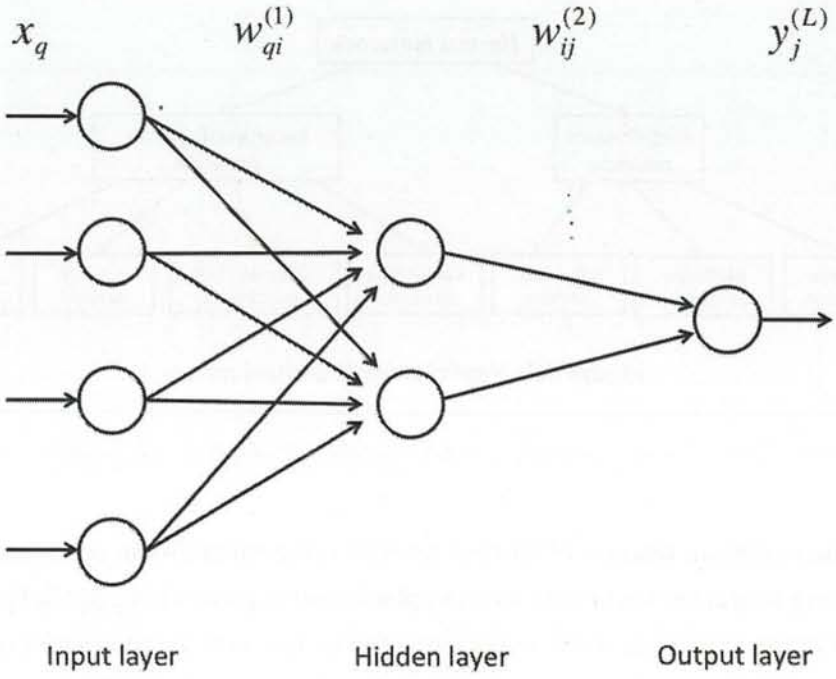



**Figure 3.8:** *Multilayer perceptron*

$$p_i(t) = \begin{cases} 1 & \text{if} h_i(t) \geq \theta, \\ 0 & \text{otherwise}, \end{cases} \tag{3.15}$$

where $\theta$ is a threshold for firing. When the neuron is fired, $R$ is subtracted from $h_i^{ref}(t)$:

$$h_i^{ref}(t) = \begin{cases} \gamma^{ref} \cdot h_i^{ref}(t-1) - R & \text{if} p_i(t-1) = 1, \\ \gamma^{ref} \cdot h_i^{ref}(t-1) & \text{otherwise}, \end{cases} \tag{3.16}$$

where $\gamma^{ref}$ is a discount rate and $R > 0$.

The presynaptic spike output is transmitted to the connected neuron according to PSP through the weight connection. The PSP is calculated as follows:

$$p_i(t) = \begin{cases} 1 & \text{if} p_i(t-1) = 1, \\ \gamma^{PSP} \cdot h_i^{PSP}(t-1) & \text{otherwise}, \end{cases} \tag{3.17}$$

where $\gamma^{PSP}$ is a discount rate and $(0 < \gamma^{PSP} < 1)$. Therefore, the postsynaptic action potential is excitatory if the weight parameter, wj,i is positive. If the condition $h_j^{PSP}(t` 1) < h_i^{PSP}(t)$ is satisfied, the weight parameter is $w_{j,i}$ trained based on the temporal Hebbian learning rule as follows:

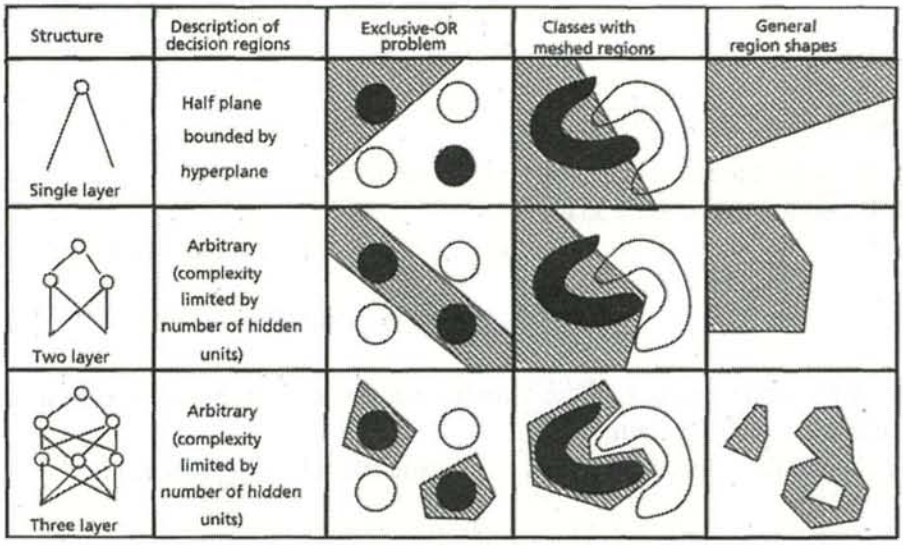



**Figure 3.9:** *Geometric interpretation of hidden layer*

$$w_{j,i} \leftarrow \tanh(\gamma^{wgt} \cdot w_{j,i} + \xi^{wgt} \cdot h_j^{PSP}(t-1) \cdot h_i^{PSP}(t)) \tag{3.18}$$

where $\gamma^{wgt}$ is a discount rate and $\xi^{wgt}$ is a learning rate.

### 3.2.6 Self Organizing Map

Unlike the previous section, unsupervised learning or sometimes also called clustering is used for grouping or segmenting data into subsets or clusters based on similarity [71]. Unsupervised learning is performed by using data without any teaching signals [99, 125, 56, 57]. K-Means and Gaussian mixture model are two of most used clustering method that using the all data in the learning phase (batch learning). Meanwhile, Self-Organizing Map (SOM) [99], Neural Gas (NG) [125], Growing Cell Structures (GCS) [56], and Growing Neural Gas [57] are some well known unsupervised learning methods that use the competitive learning approach. In the next two subsection we will discuss about two unsupervised learning used in this thesis which is Self Organizing Map (SOM) and Growing Neural Gas (GNG).

SOM was developed by professor Kohonen. SOM is a special type of competitive learning network that defines a spatial neighborhood for each output unit. SOM is often applied for extracting relationship among observed data, since SOM can learn the hidden topological structure from data. Regarding as topological structure extraction, SOM preserves the relative distance between the points while conducting mapping. Points that are near each other in the input space are mapped to nearby map units in the SOM. The SOM can thus serve as

a cluster analyzing tool of high-dimensional data. The algorithm of SOM can be written in algorithm 3.3.

**Algorithm 3.3** SELF-ORGANIZING MAP

**Step 1:** Initialize the reference vector $m_i(0)$ randomly.
**Step 2:** Present the input data $\mathbf{x(t)}$.
**Step 3:** Select the winning vector of the reference vector using the following equation

$$c = \arg\min_i ||\mathbf{x}(t) - \mathbf{m}_i(t)||, \tag{3.19}$$

where $c$ is the node number of the winning vector, $\mathbf{x}(t)$ is the input data at time $t$, and $\mathbf{m}_i(t)$ reference vector $i$ at time $t$.
**Step 4:** Update the reference vector $\mathbf{m}_i(t)$ using the following equation

$$\mathbf{m}_i(t+1) = \mathbf{m}_i(t) + h_{ci}(t)[\mathbf{x}(t) - \mathbf{m}_i(t)], \tag{3.20}$$

where $h_{ci}(t)$ is the neighborhood function. An example of neighborhood function can be shown as,

**(1)**

$$h_{ci}(t) = \begin{cases} \alpha(t) & \text{if} \quad i \in N_c \\ 0 & \text{otherwise} \end{cases} \tag{3.21}$$

**(2)**

$$h_{ci}(t) = \alpha(t) \cdot \exp\left(-\frac{||\mathbf{r}_c - \mathbf{r}_i||}{2\sigma^2(t)}\right), \mathbf{r}_c \in R^2 \quad \text{and} \quad \mathbf{r}_i \in R^2 \tag{3.22}$$

where $r$ is the node vector, $N_c$ is the neighborhood set, and $\alpha(t)$ is the learning rate.
**Step 5:** Repeat steps 2 through 4 until the convergence condition is fulfilled.

### 3.2.7 Growing Neural Gas

As discussed in the previous subsection, in SOM the number of nodes and the topological structure of the network are designed beforehand [99]. Although NG has the constant number of nodes, however its topological structure is updated according to the distribution of sample data [125]. On the other hand, GCS and GNG can dynamically change the topological structure based on the adjacent relation (edge) referring to the ignition frequency of the adjacent node according to the error index. GCS does not delete nodes and edges and it must consist of $k$-dimensional simplices whereby $k$ is a positive integer chosen in advance. On the other hand, GNG can delete nodes and edges based on the concept of ages [57]. The

initial configuration of each network is a $k$-dimensional simplex, if $k = 1$ then it is a line, if $k = 2$ then it is a triangle, and if $k = 3$ then it is a tetrahedron [56]. GCS has been applied to construct 3D surface models by triangulation based on 2-dimensional simplex. However, because the GCS does not delete nodes and edges, the number of nodes and edges is over increasing. Another disadvantage of GCS is that it cannot divide the sample data into several segments. GNG can overcome these drawbacks. When applying GNG, the distance criterion is used for extracting human motions. The GNG algorithm is described in Algorithm 1.

The following notations are used in the learning algorithm of GNG [57, 58]: $\mathbf{r}_i$ is the 3-dimensional vector of a node (reference vector, $\mathbf{r}_i \in \mathbb{R}^3$); $\mathbf{v}$ is the 3-dimensional input data, calculated from the Kinect data, describes the relative position from shoulder where shoulder position is set at $(0, 0, 0)$, $A$ is a set of node indices, $N_i$ is a set of node indices connected to the $i$-th node, and $a_{i,j}$ is the age of the edge between the $i$-th and the $j$-th node.

**Algorithm 3.4** GNG ALGORITHM

**Step 1:** Generate two units at random position, $\mathbf{r}_1$, $\mathbf{r}_2$ in $\mathbb{R}^3$. Initialize the connection set.

**Step 2:** Generate an input data $\mathbf{v}$ randomly according to $p(\mathbf{v})$ which is the probability density function of data $\mathbf{v}$.

**Step 3:** Select the nearest unit (winner), $s_1$ by Eq. (3.23) and the second-nearest unit, $s_2$ by Eq. (3.24).

$$s_1 = \arg\min_{i \in A} ||\mathbf{v} - \mathbf{r}_i|| \tag{3.23}$$

$$s_2 = \arg\min_{i \in A \setminus \{s_1\}} ||\mathbf{v} - \mathbf{r}_i|| \tag{3.24}$$

**Step 4:** If a connection between $s_1$ and $s_2$ does not exist already, create the connection. Set the age of the connection between $s_1$ and $s_2$ to zero, $a_{s_1,s_2} = 0$.

**Step 5:** Add the squared distance between the input data and the winner to a local error variable (which is initialized as 0): $E_{s_1} \leftarrow E_{s_1} + ||\mathbf{v} - \mathbf{r}_{s_1}||^2$.

**Step 6:** By using the total distance to the input data, update the reference vectors of the winner node, $s_1$ (see Eq. (3.23)) using Eq. (3.25) and its direct topological neighbors using Eq. (3.26) by the learning rate $\eta_1$ and $\eta_2$, respectively.

$$\mathbf{r}_{s_1} \leftarrow \mathbf{r}_{s_1} + \eta_1 \cdot (\mathbf{v} - \mathbf{r}_{s_1}) \tag{3.25}$$

$$\mathbf{r}_j \leftarrow \mathbf{r}_j + \eta_2 \cdot (\mathbf{v} - \mathbf{r}_j) \quad \text{if } j \in N_{s_1} \tag{3.26}$$

**Step 7:** Increment the age of all edges emanating from $s_1$: $a_{s_1,j} \leftarrow a_{s_1,j} + 1 \quad \text{if } j \in N_{s_1}$

**Step 8:** Remove edges with an age larger than $a_{max}$. If this results in units having no more emanating edges, remove those units as well.

**Step 9:** If the number of input signals generated so far is an integer multiple of a parameter $\lambda$, insert a new unit using the following steps:

a. Select the unit $q$ with the maximum accumulated error according to Step 5.

b. Add a new unit $r$ to the network and interpolate its reference vector from $q$ and $f$ using Eq. (3.27), where $f$ is that neighbor of $q$ which the largest error has according to Step 5.

$$\mathbf{r}_r = 0.5 \cdot (\mathbf{r}_q + \mathbf{r}_f) \tag{3.27}$$

c. Insert edges connecting the new unit $r$ with units $q$ and $f$, and remove the original edge between $q$ and $f$.

d. Decrease the error variables of $q$ and $f$ by a fraction $\alpha$:

$$E_q \leftarrow E_q - \alpha \cdot E_q \tag{3.28}$$

$$E_f \leftarrow E_f - \alpha \cdot E_f \tag{3.29}$$

e. Interpolate the error variable of $r$ from $q$ and $f$:

$$E_r = 0.1 \cdot (E_q + E_f) \tag{3.30}$$

**Step 9:** Decrease the error variables of all units:

$$E_i \leftarrow E_i - \beta \cdot E_i \quad (\forall i \in A) \tag{3.31}$$

**Step 10:** Continue with Step 2 if a stopping criterion is not yet fulfilled. The net size or some performance measure can be used as a stopping criterion.

---

## 3.3 Evolutionary Computing

Evolutionary computation is a field of simulating evolution on a computer. The evolutionary computation, basically divided into iterative and alternation processes of candidate solutions. Historically, the evolutionary computation are divided into three main categories: genetic algorithm by J.Holland [104] evolutionary programming by L.Fogel, and evolution strategy by H.Schwefel and I.Rechenberg. The optimization process is done by evolutionary computation through multi-point search operating on a set of individuals called population. Through selection process and genetic operation such as crossover and mutation, individuals of the population evolve toward better ones.

### 3.3.1 Simple genetic algorithm

A genetic algorithm mainly consisted of process of reproduction, recombination, and mutation. In genetic algorithm, a candidate solution called individual is encoded into a finite string called genotype. Binary coding or gray coding is often used as the genotype space. While the search is done in the genotype space, the mapping from genotype space into phenotype space is required for evaluating solution. Basically, procedure of simple genetic algorithm can be written as follows,

**Algorithm 3.5** Simple genetic algorithm

```
begin
Initialization
while Termination condition = False do
  Roulette wheel selection
  Crossover
  Mutation
  Evaluation
end while
```

One step of iteration in this procedure is called a generation. The initial population is randomly generated as candidate solutions (individuals) and each individual is assigned to an objective function their fitness. Later, selection method is used for reproducing the fitter individuals. Originally, roulette wheel is used as a fitness-proportionate selection. This method selects an individual with the following probability,

$$p_i = \frac{fit(x_i)}{\sum_{j=1}^{n} fit(x_j)} \tag{3.32}$$

where $fit(x_i)$ is the fitness value of the $i$th individual $x_i$ and $n$ denotes the population size. Other selection methods include Boltzmann selection, rank selection, and tournament selection. In this thesis we used the Boltzman selection, where the selection method is done with the following probability,

$$p_i = \frac{\exp\left(fit(x_i)/T\right)}{\sum_{j=1}^{n} \exp\left(fit(x_j)/T\right)} \tag{3.33}$$

where $T$ is temperature. The selection pressure can be adapted by updating $T$. One of disadvantage of the above stochastic selections is the best individual in a population might be eliminated. Therefore, elitism is introduced into a genetic algorithm. Elitist selection is used for maintaining the best fitted individual into the next population.

In genetic algorithm, crossover generates new individuals by exchanging the substrings between two individuals chosen randomly from a population.On the other hand, mutation changes a string by flipping some of the bits in a string at random. The generated offspring are evaluated using a fitness function or simulation-based evaluation. However, sometimes it is very difficult to design fitness function, therefore human evaluation can be directly used for evaluating candidate solutions. The method using human evaluation is called an interactive genetic algorithm. The above process is repeated until the termination condition is satisfied.

### 3.3.2 Steady State Genetic Algorithm (SSGA)

In genetic algorithm there are two reproductive technique, generational reproduction model, which is probably the most widely used, and the other is a steady-state reproduction model. The generational reproduction replaces all the population at once, while the steady-state reproduction replaces only a few individuals at once.

In steady state reproduction, only a few individuals are replaced during a generation. The crossover and mutation are also performed against only a few individuals. In this method, there are several methods for selecting an individual to be deleted. Syswerda [195] proposed three alternative deletion methods as follows,

1. Delete least fitness: deletion of the individual with the least fitness in the population.
2. Exponential ranking: The worst individual has some probability, p of being deleted. If it is not selected, then the next to the last also has p chance, and so on.
3. Reverse fitness: Each individual has probability of being deleted according to fitness value.

### 3.3.3 NSGA-II

Nondominated sorting genetic algorithm II (NSGA-II) was proposed by Deb et al. [36]. The steps of NSGA-II are as follows:

- Step 1: A random initial parent population of $N$ candidate solutions is created ($P_0$).
- Step 2: In each $t$ generation, from the $P_t$ parent population an offspring population $Q_t$ is created by selection, crossover, and mutation operators.
- Step 3: The combined parent-offspring population, $R_t = P_t \cup Q_t$ is formed with $2N$ individuals. A nondominated sorting algorithm is applied on $R_t$ which results in different nondominated fronts $F_1$, $F_2$ etc. (See Subsection III-B)

- Step 4: On the $F_1$, $F_2$ etc. fronts, another sorting procedure, the crowing distance sorting is applied. (See Subsection III-C)

- Step 5: Based on the results of the previous two sorting procedures, the best $N$ solutions are selected to form the $P_{t+1}$ population. This procedure is repeated in each generations from Step 2.

#### 3.3.3.1 Nondominated sorting

In problems with two objectives, point $I_1$ dominates point $I_2$, when all fitness functions of $I_2$ are not better or lower than $I_1$, and $I_1$ is better in at least one of the fitness functions. Figure 3.10a shows a minimization problem with two objectives. Equation (3.34) explains the mathematical concept of nondominated sorting.

$$I_1 \prec I_2(I_1 \; dominates \; I_2) \Leftrightarrow \forall i : I_1^i \leq I_2^i \wedge \exists j : I_1^j < I_2^j \tag{3.34}$$

In a multi-objective problem, we may get more than one optimal solution according to the dominance definition in Eq. (3.34). There are some non-comparable solutions which do not dominate each other. Those solutions which are not dominated by other solutions are the individuals in the so-called Pareto front. The goal of the multi-objective optimization task is to find the Pareto front. In NSGA-II, the $F_1$ front is the nondominated front, thus it contains the Pareto individuals. $F_2$ contains those individuals which are dominated only by individuals in $F_1$ front. $F_3$ contains those individuals which are dominated only by individuals in $F_1$ and $F_2$ fronts, etc.

#### 3.3.3.2 Diversity preservation

The diversity among the non-dominated solutions in NSGA-II is maintained by using the procedure of crowding comparison. This method is shown in Fig. 3.10a, between points B and D. Point D is an individual that is a better choice for removal from the population. If we remove point B, a good representative member will be absent in the large range between the A and the C individuals. However, by removing point D, the members will still remain diverse, because there are other members in the neighborhood of this point. There are good crowding distance among individuals A, B, C, and E.

Formally, the crowding distance with two objective functions at point $i$ according to Fig. 3.10b is represented by Eq. (3.35).

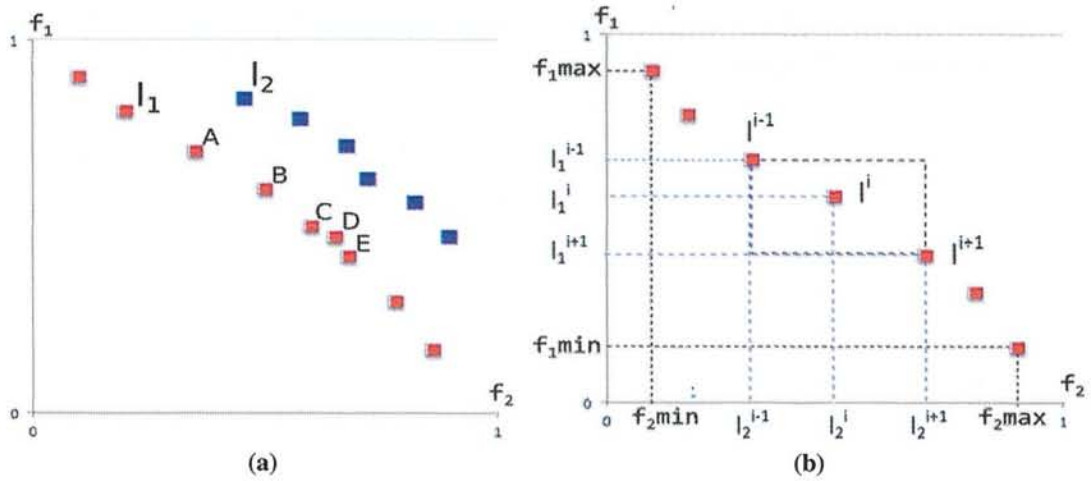



**Figure 3.10:** *(a) Supposed points in a set with equal rank (b) Crowding distance for point i*

$$I^i = \sum_{j=1}^{2} \left( \frac{|I_j^{i+1} - I_j^{i-1}|}{f_j^{max} - f_j^{min}} \right) \tag{3.35}$$

where $j = \{1, 2\}$ is the ID of objectives; $I^i$ is the proportion of the area in which point $i$ is included; $f_j^{max}$ and $f_j^{min}$ are the maximum and the minimum value of the $j$th objective in one generation, respectively; $I_j^{i+1}$ and $I_j^{i-1}$ are the $j$th objective value of the neighbors of the $i$th solution. $I^i$ indicates the crowding distance of point $i$. Therefore, the chromosome that has higher crowding distance value will cover larger area, and therefore it is preferred.

# Chapter 4

# Robot Development

In this section, we will explain the robot development that used for implementation of the proposed model. We have developed several legged robot in biped and quadruped mode. In this chapter, we also will explain our development of locomotion in conventional model as the comparison with the proposed model.

## 4.1 Humanoid Biped Robot

In the developed humanoid robot, there are 6 joints represent one leg where its kinematic analysis and coordinate transformation are represented by Denavit-Harternberg table shown in Table 4.1. Those coordinate transformations are calculated by using D-H matrix convention [39]. We create and conducted the experiments in computer simulation Open Dynamics Engine. The robot design in the simulation can be seen in Fig. 5.54b

**Table 4.1:** *Denavit-Haternberg parameters*

| $i$th Joint | $\alpha_i$ | $\theta_i$ | $\theta_i^{(0)}$ | $d_i$ | $r_i$ |
|---|---|---|---|---|---|
| 1 | $90^o$ | $\theta_1 + \theta_1^{(0)}$ | $0^o$ | 0 | 0 |
| 2 | $-90^o$ | $\theta_2 + \theta_2^{(0)}$ | $90^o$ | 0 | 0 |
| 3 | $0^o$ | $\theta_3 + \theta_3^{(0)}$ | $0^o$ | 0 | 30 cm |
| 4 | $0^o$ | $\theta_4 + \theta_4^{(0)}$ | $0^o$ | 0 | 30 cm |
| 5 | $-90^o$ | $\theta_5 + \theta_5^{(0)}$ | $90^o$ | 0 | 0 |
| 6 | $90^o$ | $\theta_6 + \theta_6^{(0)}$ | $0^o$ | 0 | 0 |

The joint structure is designed for supporting the robot to be able to perform omnidirectional movement. In order to support the proposed algorithm, several sensors are installed to

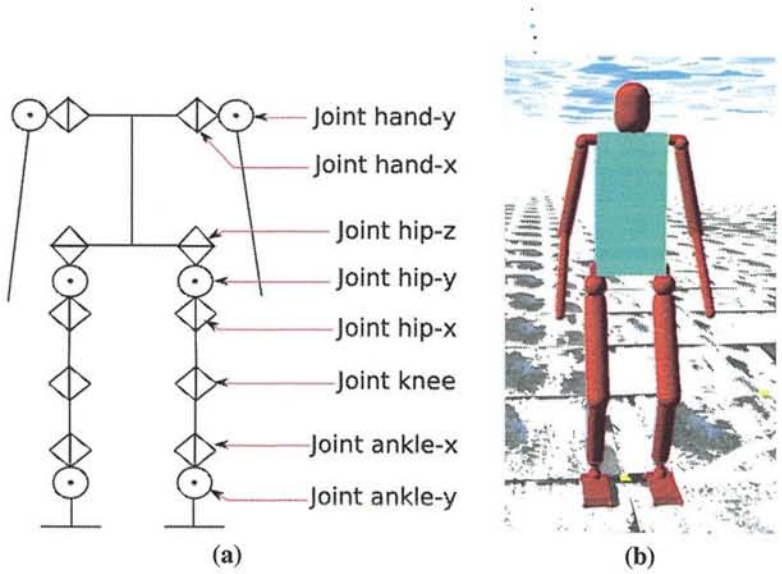



**Figure 4.1:** *a) Robot design from front side. b) Robot design from side*

the robot. Those sensors are, inertial sensor, ground touch sensor, and current sensor installer in each joint. Although we used computer simulation in the implementation, the measurements we used are possible to be applied in the real robot. From inertial sensor (inertial measurement unit), we get the body tilt of robot. Ground touch sensor provides the information whether robot feet touch the surface or not. The torque sensor in each joint implies the energy required to generate the pose of robot. Those sensors are considered in proposed push recovery controller as the control parameters.

### 4.1.1 Kubota's Simple Humanoid Robot (K-SHuBot)

n this research, we implemented the proposed system into a biped robot with 16 degrees of freedom: 4 DoF for hands and 12 DoF for feet. The robot's height is 42 cm and its weight is 2 kg. The joint structure of the robot is depicted in Fig. 7.7 and the picture of the robot is shown in Fig. 4.2. A mechanical frame from Bioloid Comprehensive Kit [6] was used for the mechanical parts of the robot and Dynamixel AX-12 [7] was used as the joint actuator.

In the hardware structure, Arduino Due [48] was used as the main controller. The robot hardware also utilized iPhone [4] as inertial sensor for feedback sensor and stabilization analyzer. The hardware structure is shown in Fig. 4.9.

The main controller computed the neural locomotion process and managed the sensor input. In order to conduct communication with the iPhone, the main controller was equipped with a Bluetooth low energy device. The iPhone sends the inertial data pitch, roll, and yaw

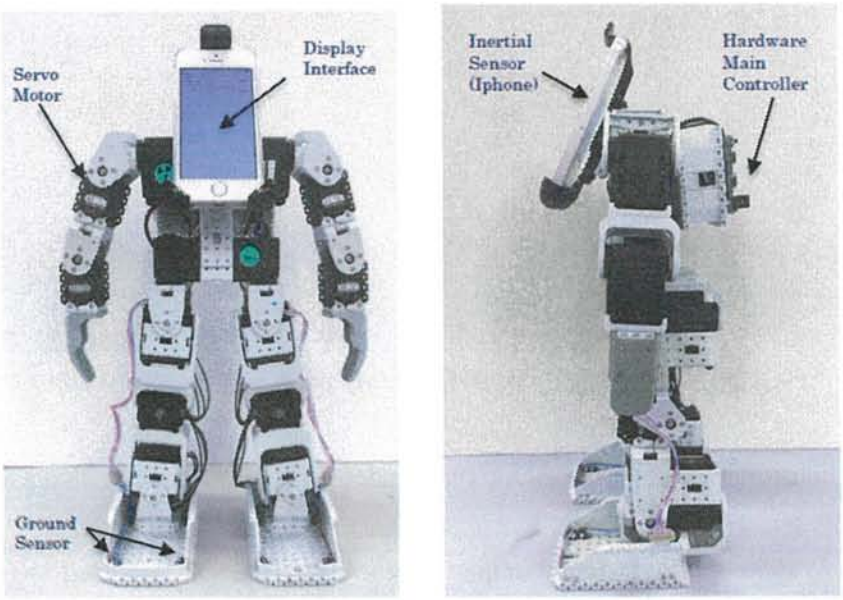



**Figure 4.2:** *Design of the real robot a) from the front; b) from the left side*

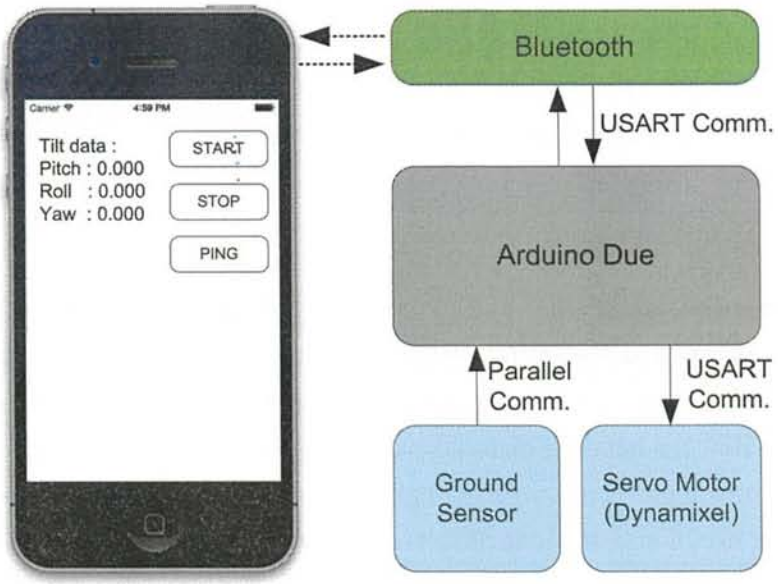



**Figure 4.3:** *Hardware structure*

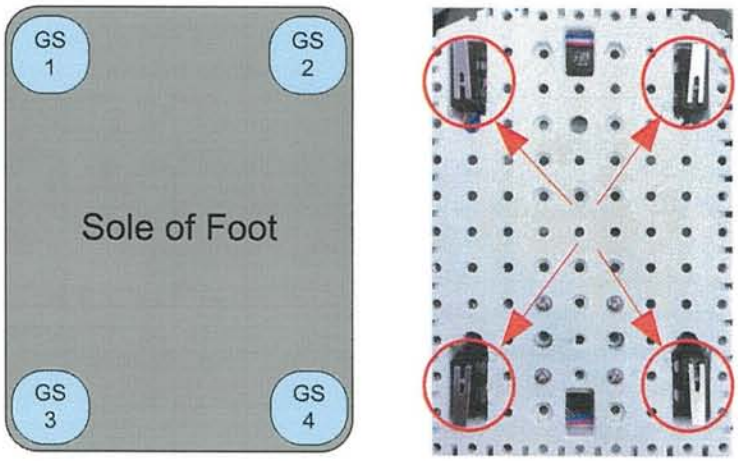



**Figure 4.4:** *Design of ground sensor*

to the main controller through the Bluetooth connection. The iPhone is also used as a user interface system.

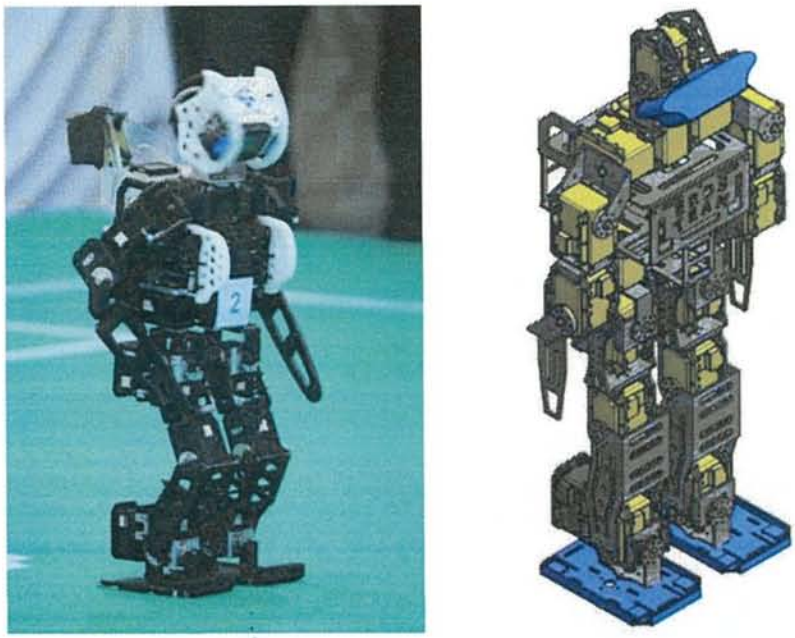
Four micro switches are installed in the four corners of the sole and used as ground touch sensors to support our system. Each sensor sends the binary data of "1", when the sensor touches the ground and "0" when the sensor does not touch the ground. The picture and the structure of the ground sensor are shown in Fig. 4.4.

### 4.1.2 EEPIS Humanoid Robot Soccer (EROS)

EROS is stand for EEPIS Robot Soccer. This robot is There are many supporting aspects which can influence the balance system when the robot does any motion, especially for robot soccer, such as mechanical system and hardware system.

#### 4.1.2.1 Mechanical Structure

EEPIS Robot Soccer or EROS has 20 degrees of freedom, 12 DoF for foot, 6 DoF for both hand and shoulder, and 2 DoF for Head. All of frames made from aluminum material. As shown at Fig. 1, we made all of frame use AutoCAD software and then we applied in CNC machine. But, for bending material we still did it manually, so it has low precision. This is still become a problem for robots balancing. All of the size in this soccer robot (the wide of feet, the size of arm, the long of feet, etc.), we made it based on the rules of RoboCup 2013s rules. But, for the feets wide, we exceed it from RoboCup 2012s standards, it is H2/32. For the foot size, we also used RoboCup 2012s standards, it is H2/30. This robot has 475 mm height and has 2900 grams weight.



Real Design Mechanical Design

**Figure 4.5:** *EEPIS Robot Soccer 3rd Generation*

All of EROSs hardware is located in center of robot body, including IMU, main controller (Fit PC), sub controller (Arm Cortex M4), and the battery also. The placing of EROSs hardware is very important, because it can influence the balancing of robots body. We have to ensure that the weight of mechanical robot, both left and right side are balance and have point of mass in the center of bodys robot. So that, we made a dynamic model and we applied the model of robot easier.

#### 4.1.2.2 Electronic Hardware Structure

We used ARM STM32f407vc as the microcontroller of this robot because it has frequency 168 MHz. If we use this microcontroller, we can access sensor data via USART easily. For accessing the data sensor (gyroscope and accelerometer) in sensor controller, we used AVR atmega8, so that the sub controller (STM32f407vc) can access sensor data via USART easily. In Fig. 2 , the hardware system has two controllers; they are namely sub controller and main controller. Sub controller has function to control servo motor controller with half duplex serial, and process sensor data that sent by sensor controller. Data of sensor was process by sensor controller firstly to eliminate the noise so that the data received by the sub controller is a good data. Main controller, has function as a processing image from camera and make some artificial intelligent for robot soccer. Main controller sends a command to

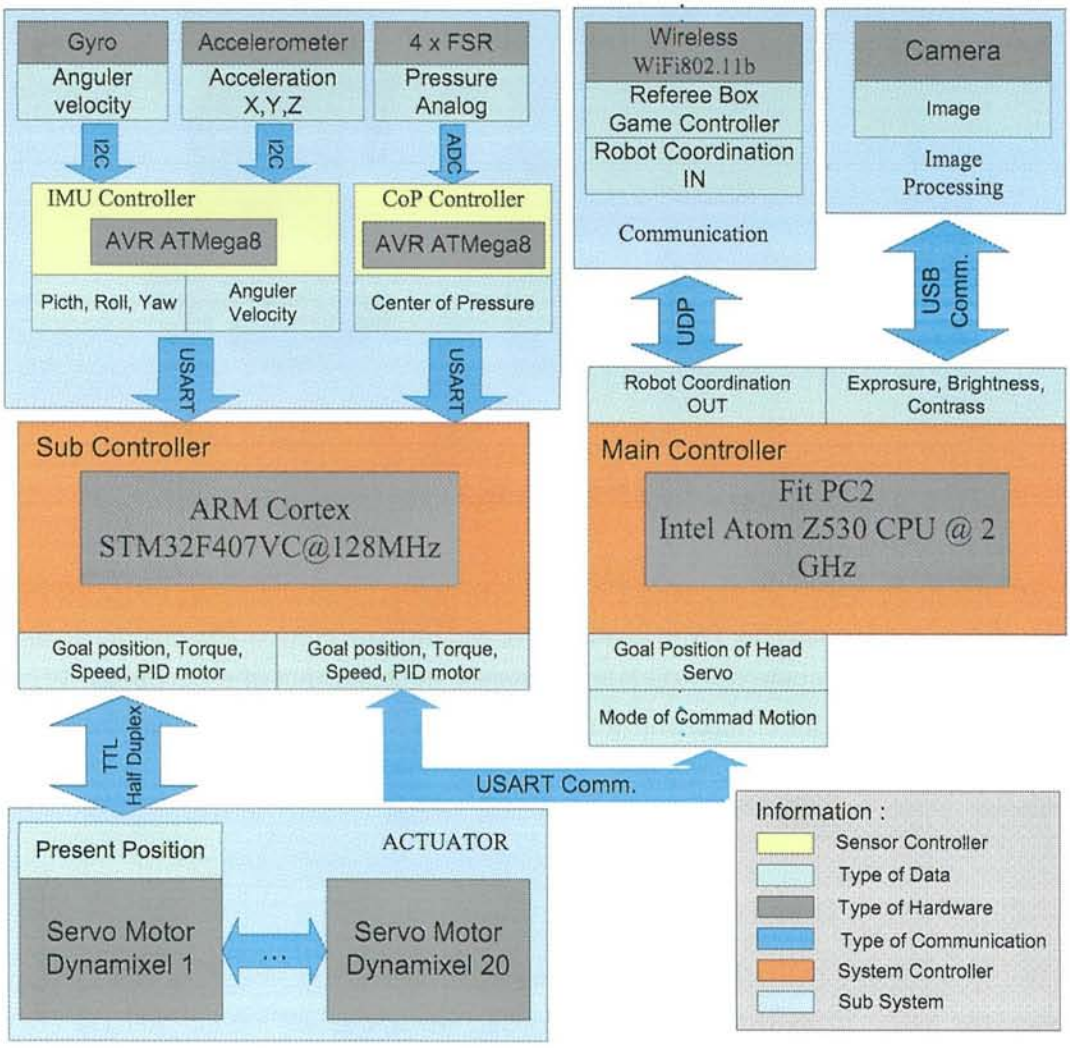



**Figure 4.6:** *Hardware construction*

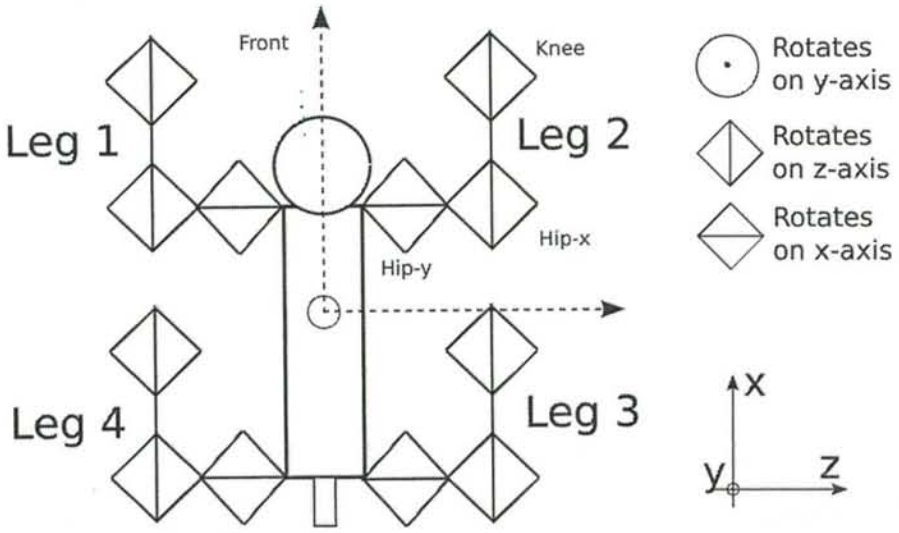



**Figure 4.7:** *Joint structure of the robot*

the sub controller to perform a particular motion. For communication between each robot to other robot and robot to referee box, we use Wi-Fi communication 2,4 GHz.

## 4.2 Quadruped Robot

### 4.2.1 Kubota's Simple Quadruped Robot (K-SQuBot)

In this research, we developed a 4-legged robot with 12 degrees of freedom, where there is 3 degree of freedom (Hip-x, Hip-y, and Knee joint) in each leg. The robot's size is approximately 30 cm x 18 cm X 20 cm, and its weight is 2 kg. The joint structure of the robot is depicted in Fig. 7.7 and the picture of the robot is shown in Fig. 7.8. A mechanical frame from Bioloid Comprehensive Kit was used for the mechanical parts of the robot and Dynamixel AX-12 was used as the joint actuator.

In the hardware structure, Arduino Mega was used as the main controller which only has 16 MHz processor frequency. For low cost development, the robot hardware also utilized iPhone as inertial sensor for feedback sensor and stabilization analyzer in order to change the sensor unit. The hardware structure is shown in Fig. 4.9.

The main controller computed the neural locomotion process and managed the sensor input. In order to conduct communication with the iPhone, the main controller was equipped with a Bluetooth low energy device. The iPhone sends the inertial data pitch, roll, and yaw to the main controller through the Bluetooth connection. The iPhone is also used as a user interface system. Another iPhone function is to be enable remote control to control the walk-

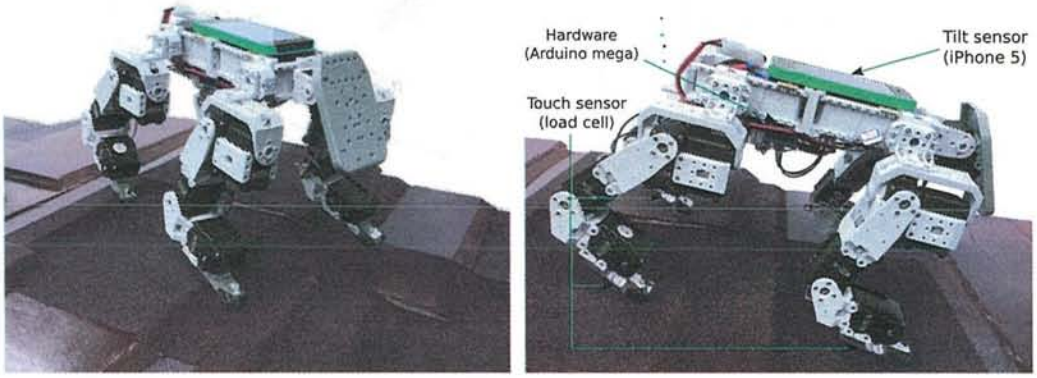



**Figure 4.8:** *Design of the real robot a) from the front; b) from the left side*

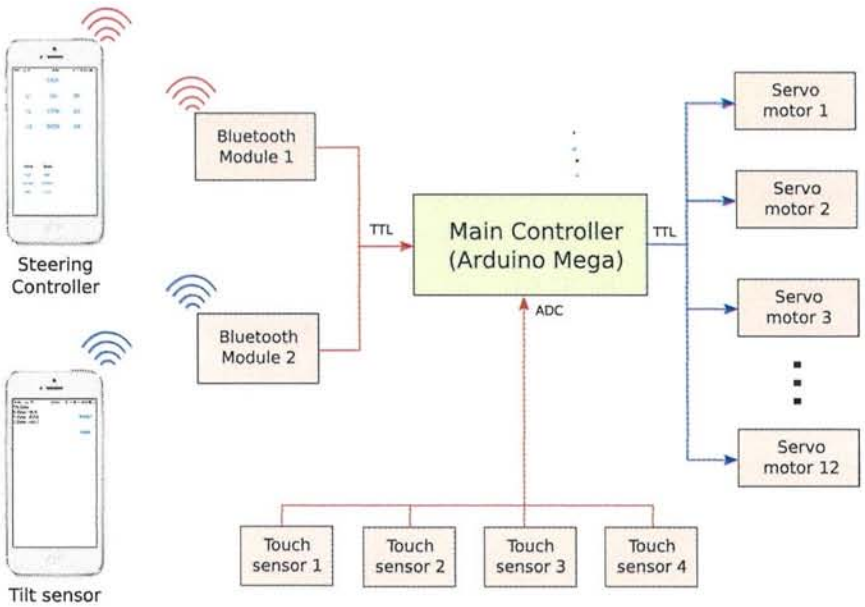



**Figure 4.9:** *Hardware structure*

ing direction and walking speed. In order to detect the ground touch when stepping, four load cells are installed in each leg. Each sensor sends the analog signal that represents the force value. When the sensor touches the ground, then the impulse signal from load cell is received by microcontroller.

## 4.3 Conventional Model Development

The aim of this development is to creating the motion generator based on conventional model for comparing with proposed alternate model.

### 4.3.1 Motion Pattern Generation

Humanoid Robot Development has increased significantly. In locomotion humanoid robot, the major problem is in the stability part. Robot should be robust that the disruption from the outside. Robot should be able to run, walk, kick the ball, and play soccer as human do. the goal of this research is how to make the stability when single support phase (SSP) the condition where robot just has one leg to support the body of robot and make the trajectory that adjust the environment surface (synthetic grass). Many methods have been studied in the balancing of humanoid locomotion. There are many methods to solve this problem. To achieve more robust motion control in these environments, the robot must be able to adjust its walking state (center of mass motion) and step length to accommodate whatever path planning approach is used [206, 229, 102, 219]. The goal of this research is make the humanoid locomotion that can be adaptive with the environment. One idea is to use the dynamical relationship between the inverted pendulum and the center of gravity (COG) approach. In our previous research we used ZMP approach. We combined the angular velocity control and pose control. We have proposed the method that controls the trajectory planning through the ZMP analyzing and tilt of robot in real-time [176]. The proposed method in this developement extends the trajectory system as the part of stability system in robot movement. By analyzing the ZMP with closed form functions, a simple and direct means to control the humanoid dynamics is realized.

This development conducted by PENS and Tokyo Metropolitan University. The development of humanoid robot has entered the 4rd years, starting in 2010. And we already finished developing previous version of EROS: EROS-I in 2010 – 2012 and EROS-II in 2012 – 2014. EROS II use open loop system in its motion system. So we can call this robot using "Offline pattern generation". This system is suit-able only with one kind of surface. Referring to the previous EROS development, the stability of the robot needs to be improved so that the os-

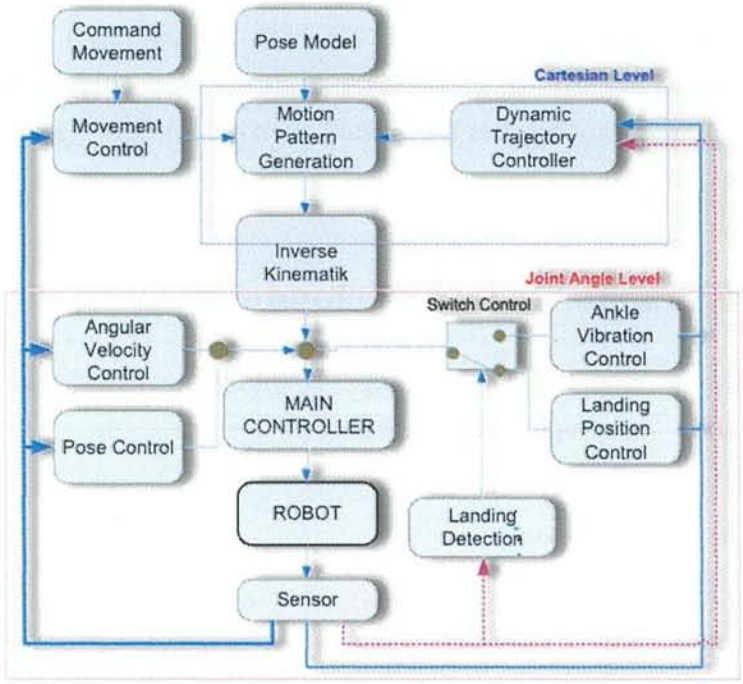



**Figure 4.10:** *Overall Control System Diagram*

cillating movement of the robot decreases. Now, we develop the new platform of EROS III. The robot is higher than before and heavier than before. In this research we develop the motion pattern generation using sine, cosine, linear function and 3rd polynomial equation that combined with hand learning system to support the stabilization. We analyze the trajectory pattern of HUBO robot [96] that was not adaptive with center of gravity condition. So we develop the dynamic step trajectory that adaptive with the environment surface condition. That uses inverted pendulum approach. We develop the dynamic pattern of trajectory. So, robot can suitable with different surface and uneven surface. We can adjust the direction, the velocity of steps trajectory. This system is affected by the inclination condition of robot body, normal force reaction that caused by foot touches to the ground. To support this system, we also design the pressure point sensor which using 4 sensors FSR. This sensor installed in the bottom of foot. By using this sensor, we can detect the force reaction and the pressure point.

#### 4.3.1.1 Design of System Control

In robot EROS there are several sub-controls that support the stability of the EROS movement. It is supported in Cartesian level with changing the Cartesian trajectory and joint angle level with changing the angle in several actuators. In the movement system of robot EROS, is firstly forming the modeling walking pose. Then the out of walking pose, is forming a

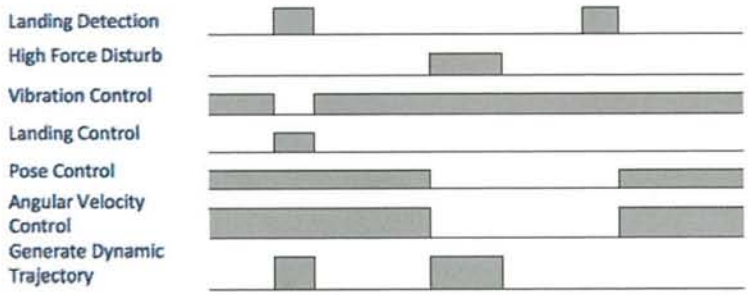



**Figure 4.11:** *Time scheduling model*

robot trajectory step by using several equations, such us sine, cosine, circle equation, as well as 3-degree polynomial equation.

In this system, there is movement control that affecting the trajectory pattern generation by regulating changes in speed and direction of motion of the robot. And also influenced by dynamic Trajectory control that affect the shape of the trajectory path by looking at the center of gravity of the robot. These controls also cut the track conditions when landing trajectory detection detect that foot has touched the ground surface. This system is located at Cartesian trajectory level. This pattern will be changed to the form of joint angle level, through a process of inverse kinematic. To get 6 Degree of Freedom Inverse Kinematics, we use combination between trigonometry equation and Denavit Hartenberg Model. In the joint angle level, there are vibration control to reduce vibration ankle due to foot swing and landing position control ankle foot position when tread. Both of these controls work interchangeably depending on the command of the landing detection. Landing detection system reads the pressure sensor placed on the soles of the feet. To support the balancing of robot body, there is combining stabilization control (pose control and angular velocity control) that reads the tilt of robot body and the angular velocity of the robot body. There is the main controller which regulates the incorporation of each control.

Figure 4.11 shows the schedule activation controller. When robot walking normally uses vibration control robot uses controller whose pelvis joint become the actuator, pose control and angular velocity control whose ankle joint become the actuator. When landing detection active, they are deactivated, changed with landing control. When robot get high force disturbance, robot generate the dynamic trajectory control and do not activate pose control and angular velocity control until landing detection active.

To increase the stability, we make addition stabilization control. We tried to combine this control system with hand reaction learning system. This system compensated the vibration and disturbance by using hand reaction. Robot learns how to use hand for stabilization in the movement of humanoid robot. This system uses recurrent neural network backpropagation

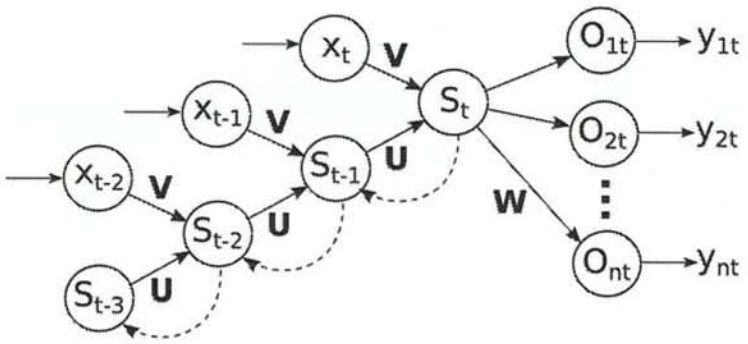



**Figure 4.12:** *Proposed RNN strcuture*

through time and installed gyroscope pitch and roll as the feedback sensor and as input unit in robot. We use hand actuator as the response of action $O_n(t)$ $i$ joint angle level that resulted from learning process . From the reaction of output layer, we get the response sensor that become the next input in RNN $s_y(t)$.

$$s_y(t) = f(ns_y(t)) = f\left(\sum_i^l x_i(t)v_{ij} + \sum_h^m s_h(t-1)u_{ij} + b_j\right) \tag{4.1}$$

$$y_k(t) = f(ny_k(t)) = f\left(\sum_j^m s_j(t)w_{kj} + b_k\right) \tag{4.2}$$

We have 2 the number of input $l$, 4 the number hidden layer $m$, and 4 the number of output layer. We need to learn $V$, $U$, and $W$ as the weight among neuron.

$$\delta_k = (d_k - y_k)f'(ny_k) \tag{4.3}$$

$$\delta_j = \sum_k^n \delta_k w_{kj} f'(ns_j) \tag{4.4}$$

where $\delta_j$ is error propagation in output node and $\delta_j$ is error propagation in hidden node. $d_k$ is the desire output. In this case, the desire output is the condition where robot has can maintain the angular velocity of robot become zero.

$$\mathbf{W}(t+1) = \mathbf{W}(t) + \eta\mathbf{s}(t)\delta_k^T \tag{4.5}$$

$$\mathbf{V}(t+1) = \mathbf{V}(t) + \eta\mathbf{x}(t)\delta_j^T \tag{4.6}$$

$$\mathbf{V}(t+1) = \mathbf{U}(t) + \eta\mathbf{s}(t-1)\delta_j^T \tag{4.7}$$

For BPTT, the error propagation is done recursively. The updated weight equation can be

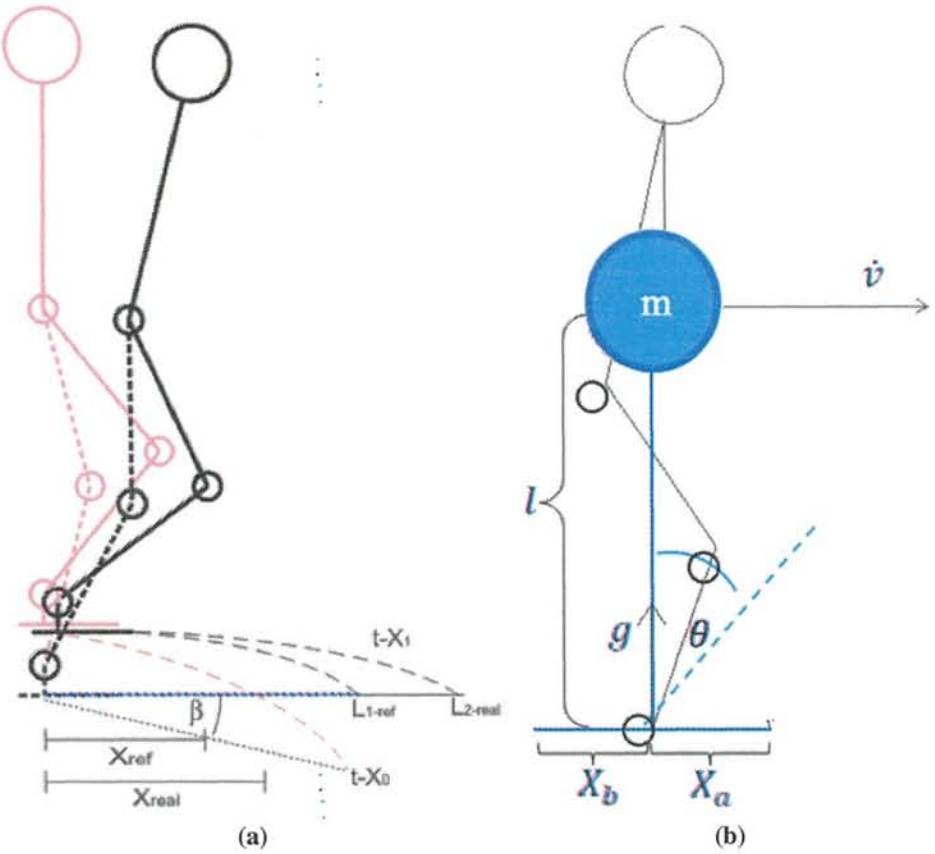



**Figure 4.13:** *(a) Changing movement pattern x-direction (b) Inverted pendulum model*

shown in Eqs. (4.5), (4.6), and (4.7).

#### 4.3.1.2 Design of Motion Pattern Generation

In this section we will explain the motion trajectory that can be changed directly depending on the robot condition. The pattern mechanical support (servo motor, hardware, and mechanical dimension) is required to support the locomotion. This system analyzes the center of gravity point from robot body based on inverted pendulum approach. Robot will position the foot step depend on the location of center of gravity point.

In Fig. 4.13, the pendulum model in walking will be shown. The acceleration $\dot{v}$ will be resulted in this model. Inverted pendulum at the vertical direction will be resulting angular acceleration $\ddot{\theta}_g = (g/L)\sin\theta$, and angular acceleration on horizontal direction is $\ddot{\theta}_z = -(\dot{v}/L)\cos\theta$. Transfer function $G(s)$ was explained in equation 2 and $\beta$ is the real angle of robot tilt.

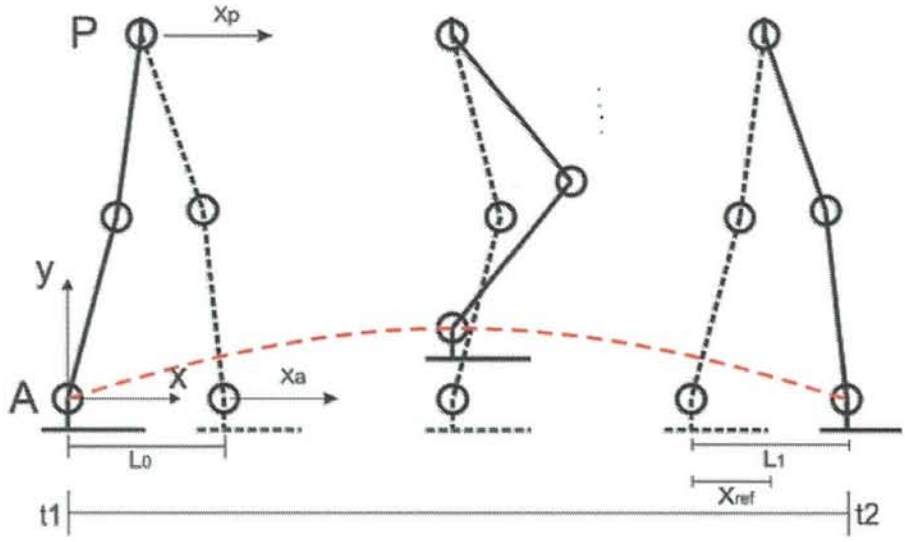



**Figure 4.14:** *Trajectory planning in x-direction*

$$G(s) = \frac{\theta(s)}{x(s)} = \frac{-s^2}{Ls^2 - g} = \frac{-s/g}{(\tau_L s + 1)(\tau_L s + 1)} \tag{4.8}$$

In Eq. (4.8), $\tau_L = \sqrt{l/g}$ is the constant time. The output of this equation got ideal angle when robot walking and becomes the angle reference of the pose control. Figure 4.13a shows the trajectory changing in $x$-direction. $t - X_0$ is the reference trajectory and $t - X_1$ is real trajectory after changed. Figure 4.14 shows trajectory planning in x-direction. When robot is walking, we define the value of step length $(L_{1-ref})$ by using DSP (Double Support Phase) center point to become center of gravity reference $(L_{1-ref} = 2x_{ref})$. We keep it with adjusting the angle body of robot. We will not change the length of step when the center of gravity point is in supported area.

$$x_{real} = \frac{g \times sin(\theta_{real})}{\dot{v} \times l} \tag{4.9}$$

Eq. (4.9) shows the real of center of gravity, where $\theta_{real}$ is the angle of robot tilt and $\dot{v}$ is the robot acceleration in horizontal direction that read from IMU sensor. When robot get high disturbance, robot will change the length of steps, depend of the condition of robot. S is difference between $x_{real}$ and $x_{ref}$. When $S > X_a$ or $S < X_b$ robot will change the next length of step. This model is also used in z-direction (coronal view). We can see Figure 4.14, where $\gamma_{real}$ is the angle of robot tilt. Robot will change the value of $E_l$ and $E_r$ when center of gravity in z-direction is not in supported area. We read this condition when the trajectory will started and we define all of the dynamic parameters, coronal view and sagittal view.

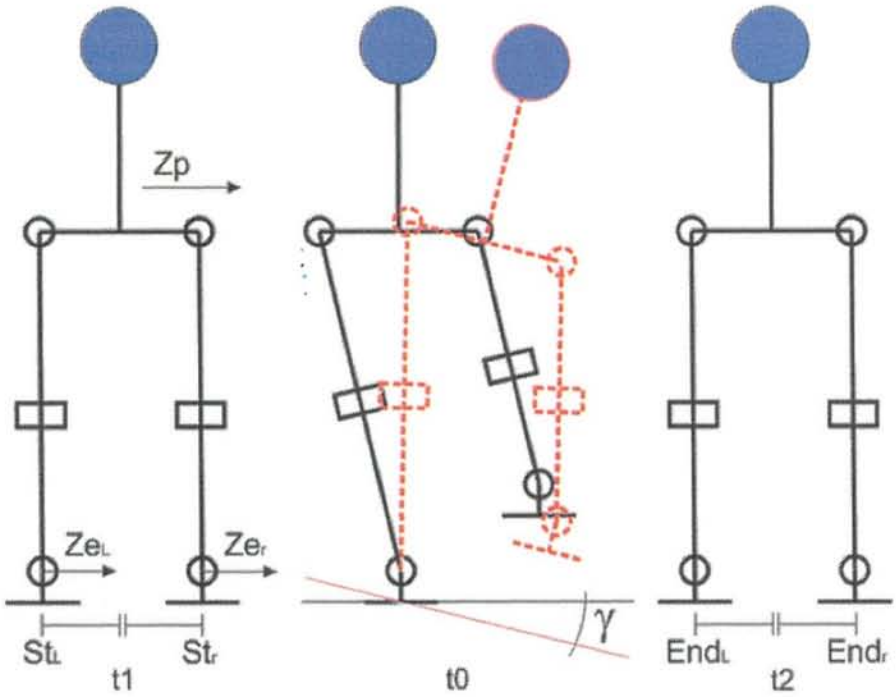



**Figure 4.15:** *Trajectory plan in z-direction*

Trajectory patterns divided by 3 pattern: trajectory x-direction, trajectory y-direction, and trajectory z-direction. In x-direction, robot has 2 trajectory equations, $\bar{x}_p$ for trajectory pelvis and $\bar{x}_a$ for trajectory ankle.

$$\bar{x}_a(t) = (L_0 + L_1)\left[\frac{t-t_1}{t_2-t_1} - \frac{1}{2\pi}\sin a\left(2\pi\frac{t-t_1}{t_2-t_1}\right)\right] cos a\beta - L_0 \tag{4.10}$$

$$\bar{x}_p(t) = \sum_{i=0}^{3} a_i \left(\frac{t-t_1}{t_2-t_1}\right)^i \tag{4.11}$$

Eq. (4.11) is 3rd order polynomial, where $\bar{x}_p(0) = -b/2$, $\dot{\bar{x}}_p(0) = ab$, $\bar{x}_p(1) = f/2$, $\dot{\bar{x}}_p(1) = af$. To get flexibility curve, $a_i$ is resulted from Gauss Jordan calculation from $0 = (t_1)$ and $1 = (t_2)$.

Figure 4.15 show the trajectory ankle in z-direction that 2 step trajectory, first trajectory equation from $t_1^1$ to $t_2^1$, when DSP condition, to SSP condition. And second trajectory equation from $t_1^2$ to $t_2^2$, when SSP condition to DSP condition.

$$Ze_l(t) = \frac{1}{2}\Big(\frac{E_l}{2}\Big[1-\eta-St_l\Big]\Big)\Big(1+A(t)\Big) + St_l \tag{4.12}$$

$$Ze_r(t) = \frac{1}{2}\Big(\frac{E_r}{2}\Big[1-\eta-St_r\Big]\Big)\Big(1+A(t)\Big)+St_r \tag{4.13}$$

$$A(t) = \begin{cases} \cos\left(\pi\frac{t-t_1^1}{t_0^1-t_1^1}\right) & \text{if } t_1 < t < t_0 \\ \cos\left(\pi\frac{t-t_0^2}{t_2^2-t_0^2}\right) & \text{if } t_0 < t < t_2 \end{cases} \tag{4.14}$$

Where $E_l$ is the goal coordinate on the left foot and $E_r$ is the goal coordinate on the right foot. $St_l$ and $St_r$ is the last coordinate in left and right foot. We can adjust $\eta$ value to get the ratio of steps.

$$\bar{z}_p(t) = \sum_{i=0}^{3} \bar{a}_i \left(\frac{t-t_1}{t_2-t_0}\right)^i \tag{4.15}$$

$$\bar{z}_p(t) = \sum_{i=0}^{3} \bar{a}_i \left(\frac{t-t_0}{t_2-t_0}\right)^i \tag{4.16}$$

Eqs. (4.15) and (4.16) are the 3rd polynomial equation. In trajectory pelvis in z-direction, the pattern is different with trajectory in x-direction. This pattern has 2 equations in 1 step that are $t_1 < t < t_0$ and $t_0 < t < t_2$. Where in first equation defined $\bar{z}_p(t_2) = 0$, $\dot{\bar{z}}_p(t_2) = -a_1$, $\bar{z}_p(t_0) = -S_y$, and $\dot{\bar{z}}_p(t_0) = 0$. And second step is defined $\bar{z}_p(t_1) = 0$, $\dot{\bar{z}}_p(t_1) = -a_1$, $\bar{z}_p(t_0) = -S_y$, and $\dot{\bar{z}}_p(t_0) = 0$.

Trajectory in y-direction has several equations for adjusting the high of steps $h$. In Eqs. (4.18) and (4.19), $Y_x(t)$ is the trajectory that effected by trajectory x-direction, and $Y_y(t)$ is trajectory that effected by trajectory z-direction. The output of the trajectory in y-direction is the calculation between $Y_x(t)$ and $Y_y(t)$

$$Y(t) = \left[2 - \left[cosa\left(2\frac{t-t_1}{t_2-t_1}\right)+1\right]\right]\frac{h}{2} \tag{4.17}$$

$$Y_x(t) = Y(t)\cos a\beta + \bar{x}_a(t)\sin\beta \tag{4.18}$$

$$Y_y(t) = Y(t)\cos a\gamma + (Z_e(t) + St_l + St_r)\sin\gamma \tag{4.19}$$

When the sensor of pressure not grounded until the end of step of time, y-trajectory will continue the swing of foot to go down and the other will support the swing of leg to go down. System will wait sensor detected the ground first, before starting the new trajectory. When landing detection active, system will stop the trajectory and starting next step trajectory. This allows different periods in each step.

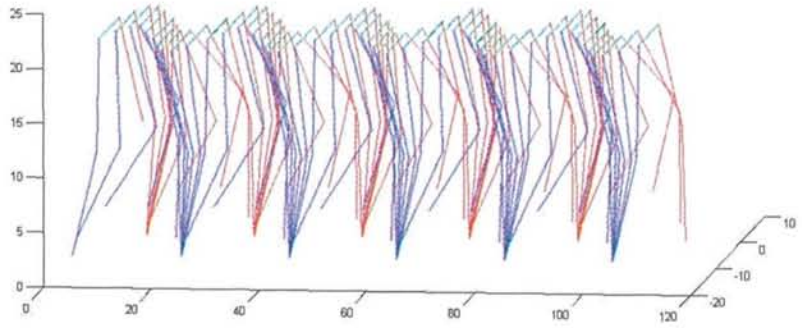



**Figure 4.16:** *Trajectory plan in z-direction*

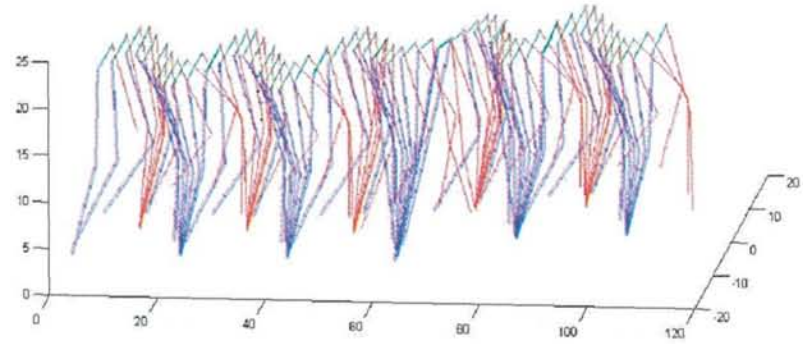



**Figure 4.17:** *Trajectory plan in z-direction when get disturbance*

#### 4.3.1.3 Experiments

In the experimental result, firstly we simulate this system in Matlab. We tried this system in normal condition when robot walking with 20 length of stride that can be seen in Fig. 4.17 and when robot got disturbance from behind that showed in Fig. 4.18. Robot will change the trajectory when get disturbance that effected shifting the ZMP out of supported area. This system is applied in EEPIS Robot Soccer that has been developed in 3 years. Figure 4.17 shows the ZMP trajectory when robot walking normally with normal speed of movement, without additional disturbance. And Fig. 4.18 shows the ZMP trajectory when robot walking and get the disturbance from behind. We analyze ZMP by read the inertial measurement unit which inside body of robot. When walking normally, ZMP robot inside the supported area of foothold. Although there are several ZMP outside the supported area, system not give the respond to change the trajectory because the distance between ZMP and supported area is slight. In Figure 4.17, the robot walking tends to the right. The position of ZMP is more in the right area from local position in robot. We can analyze that the mechanical structure of robot is not balance. Right side is heavier than left side. When robot get high disturbance from behind, trajectory of robot will change trajectory following the direction of disorder

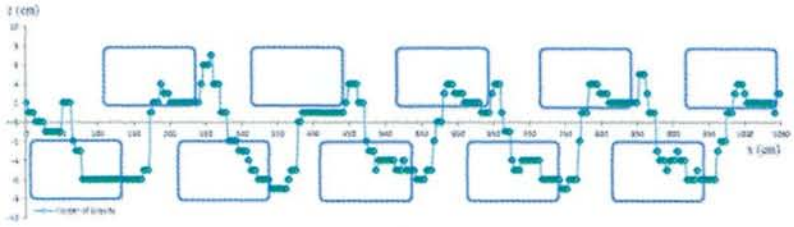


**Figure 4.18:** *ZMP Trajectory normally movement*

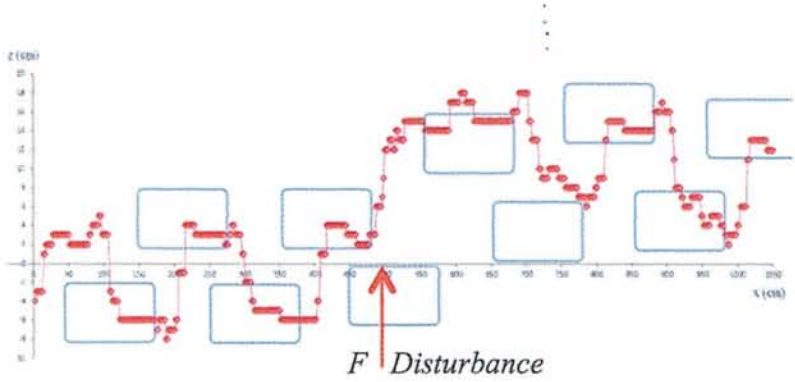



**Figure 4.19:** *ZMP Trajectory with disturbance*

force.

We read trajectory in y-direction 2 conditions, we can compare the differences when walking robot get disturbance and without disturbance. Figure 4.19 shows the trajectory z-direction in normally walking. Robot walks in flat surface. Figure 4.20 is when robot walking in uneven surface. Robot can keep stability with dynamic trajectory and dynamic step period. Figure 4.21 showed normal walking that has normal surface. By using this system, trajectory of robot can suitable in the real condition of robot and can adapt time start and time finish that showed in Fig. 4.22.

This control system can reduce the size of soles of the feet from $h^2/32$ become $h^2/36$. This control system will be developed until be able to support the robot that has soles of the feet size same as the human have $h^2/88$ until 2050. The walking pattern still doesn't as smooth as human walking, because in this robot has limited degree of freedom. so, robot can not support all human walking movement. we try to adapt the human movement as similar as it can.

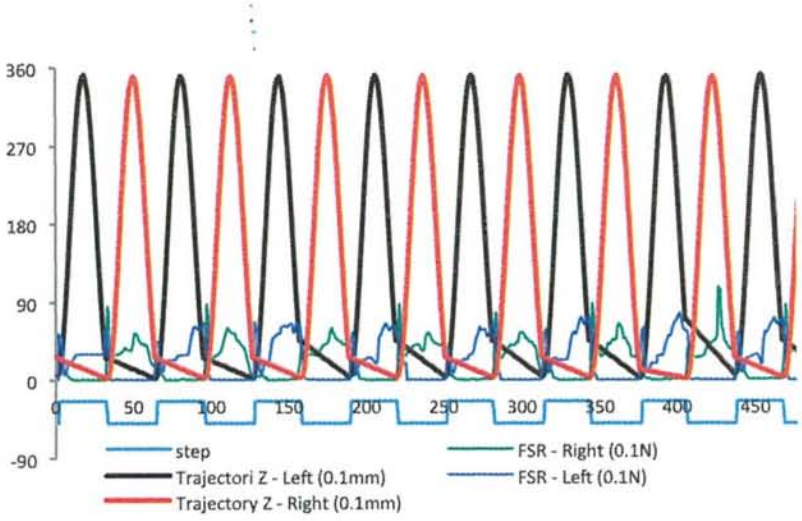



**Figure 4.20:** *Trajectory z-direction in flat surfaces*

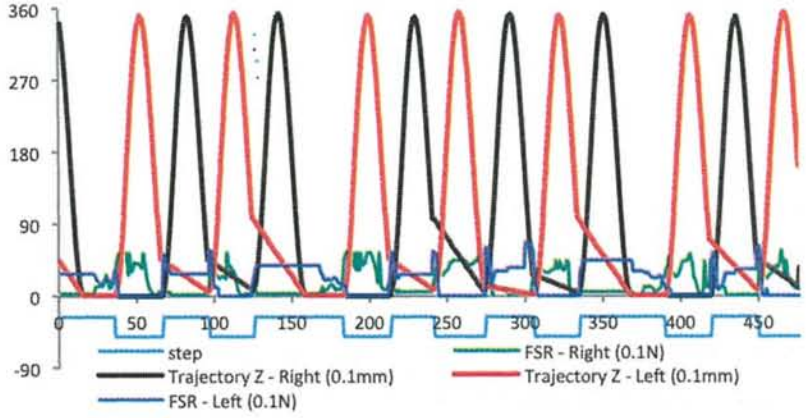



**Figure 4.21:** *Trajectory z-direction in uneven surfaces*

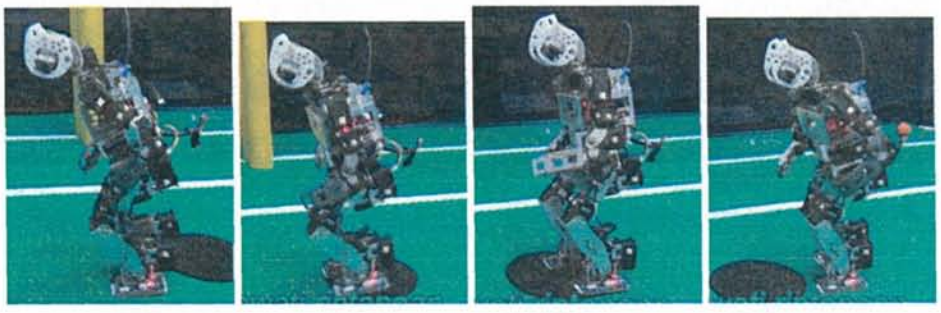


**Figure 4.22:** *Robot moves on uneven terrain*

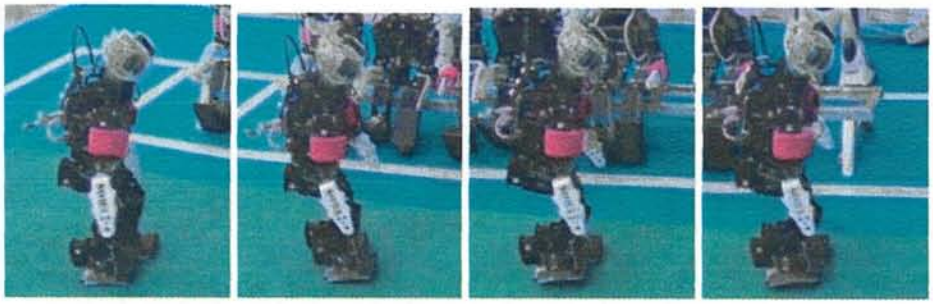


**Figure 4.23:** *Robot moves on artificial grass*

## 4.3.2 Stability Model

### 4.3.2.1 Introduction

Instability is one of the major defects in humanoid robots. Various methods on the stability and reliability of humanoid robots have been actively studied. Among the research challenges specific to the Humanoid League is maintaining the dynamic stability of the bipedal robots while they are walking, running, kicking, and performing other tasks. Another example is the coordination of perception (with a human-like limited field of view) and locomotion. The humanoid soccer robots must also be robust enough to deal with challenges from other players. Many of the existing methods for humanoid walking control are constrained by the need to regulate foot positions without any consideration for navigation of the humanoid within a complex environment. To achieve more robust motion control in these environments, the robot must be able to adjust its walking state (center of mass motion) and step length to accommodate whatever path planning approach is used [206], [219, 54, 220, 21, 92, 189, 190, 204, 93, 120].

The goal of the present developement is to provide humanoids with high-mobility, developing a realtime motion generation method, positively using dynamics of robots. An idea is to use the dynamical relationship between the Zero Moment Point (ZMP) and the center of gravity (COG). The legged system has a similar dynamics to that of inverted pendulum, whose supporting point is equivalently located at the ZMP. We have proposed the method that controls the COG of the whole humanoid body system in realtime through the ZMP manipulation. The proposed method in the previous developement extends the conventional 3 Dimensional Linear Inverted Pendulum Model (3D-LIPM) in such a way so that feasibility of desired motions is considered. By manipulating the ZMP with closed form functions, a simple and direct means to control the humanoid dynamics is realized. It is similar with the method proposed by Sugihara [189, 190] in the usage of analytical solution of simplified equation of motion. However, we enable control of CoM (Center of Mass) velocity and

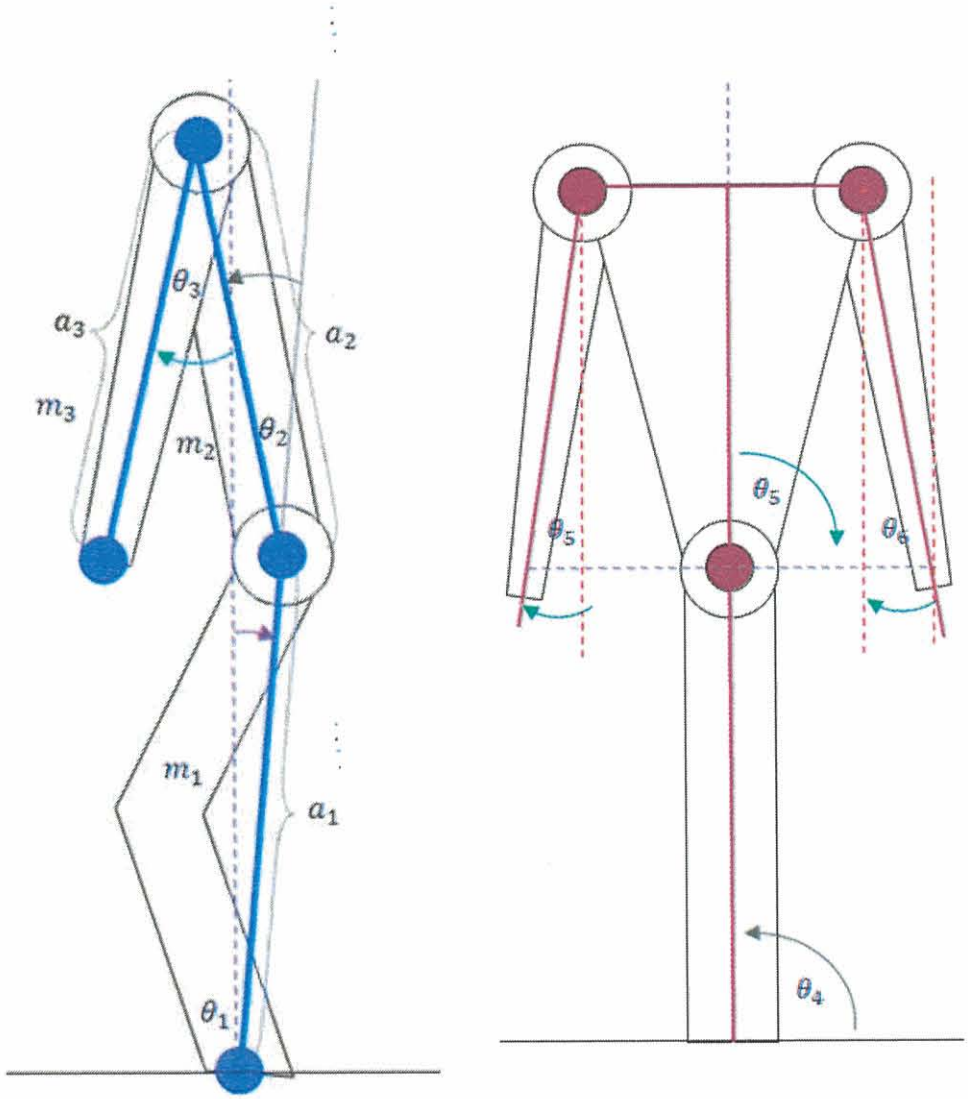



**Figure 4.24:** *Dynamic model of proposed Robot*

step lengths directly and independently. We have used this with the implementation of fuzzy inference system for walking and turning tap movement and we combine with control of vision system for humanoid soccer using camera [192, 191]. In soccer, the robot is required to be able to perform basic movements for playing football. We applied at RoboCup 2012 at Mexico and RoboCup 2013 at Netherlands. In this development, we proposed robot basic movements that currently have less speed action. The robot must be able to run fast to catch the ball, kickball and get up quickly. We can examine the attitudes of human balance to help us understand the problems that arise in balancing a soccer robot.

The complexity of the human nerve system generates a feedback from a variety of sensor systems. The use of information consisting of pressure on the foot, given external force, visual and inertia, the human brain is capable of processing a wide range of joint movement and structured to maintain stability. The integration between sensors, balance and control systems required in order to fit the human body is very difficult. The system design involves counting perfect balance and controls are very thorough. The partial implementation of the system of the human body balance using the body's control poses and angular velocity control, which can reduce the moment of the force generated when suddenly stopping and running.

#### 4.3.2.2 Modeling and Controlling System

**4.3.2.2.1 Modeling System** Fig. 4.24 shows dynamic model of EROS with its pitch position and roll position. By using the modeling position shown at Fig. 4.24, the robot can be easily controlled. This modeling divided into three parts, there are hands robot, bodys robot, and foots robot. Each part has different weight, and has their own elevation angles that has been influenced by Inertia Moment (I) of each part, and the Damper (D) has been influenced by level of strain of each joint (K). The formula for the transfer kinematic function can be seen below:

$$\tau_1 = [J_1 s^2 + (D_1 + D_2)s + (K_1 + K_2)]\theta_1 - [D_2 s + K_2]\theta_2 \tag{4.20}$$

$$\tau_2 = -[D_2 s + K_2]\theta_1 + [J_2 s^2 + (D_2 + D_3)s + (K_2 + K_3)]\theta_2 - [D_3 s + K_3]\theta_3 \tag{4.21}$$

$$\tau_3 = -[D_3 s + K_3]\theta_2 + [J_3 s^2 + (D_3 + D_4)s + (K_3)]\theta_3 \tag{4.22}$$

Eq. (4.20) is the formula of torsi that needed ($\tau_1$) for joint 1 ($\theta_1$), and Eq. (4.21) is the formula of torsi that needed ($\tau_1$) for joint 2. Eq. (4.22) is the formula of torsi that needed ($\tau_1$)

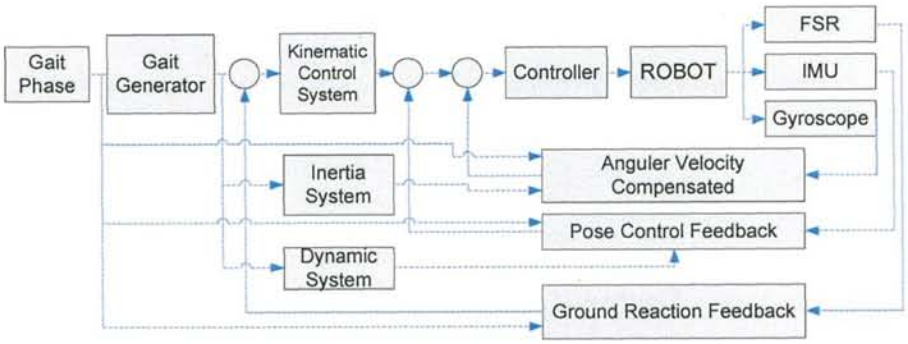



**Figure 4.25:** *Stability system of proposed Robot*

for joint 3. If we combined those three formula, it will become Eq. (4.23), by using Gauss Jordan elimination, we can get the angles in joint 1, 2, and 3.

$$\begin{pmatrix} G_1 & G_2 & 0 \\ G_2 & G_3 & G_4 \\ 0 & G_4 & G_5 \end{pmatrix} \begin{pmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \end{pmatrix} \tag{4.23}$$

where:

$$G_1 = [J_1 s^2 + (D_1 + D_2)s + (K_1 + K_2)]$$

$$G_2 = -[D_2 s + K_2]$$

$$G_3 = [J_2 s^2 + (D_2 + D_3)s + (K_2 + K_3)]$$

$$G_4 = -[D_3 s + K_3]$$

$$G_5 = [J_3 s^2 + (D_3 + D_4)s + (K_3)]$$

Fig. 4.25 describes the process of EROSs control system mode. The kinematic system built by using lookup data table as the result of kinematic pattern counting.

The robot uses feedback to support the robot's stability. Angular velocity compensated serves to compensate the sway like pendulum that obtain oscillated angular speed. Feedback Pose Control serves to maintain the robot's gravity center, so it will always in the stable condition when the robot runs. The ground reaction serves to reduce the buffeting feet when the robot walks or runs.

This development focuses on angular velocity feedback and feedback of pose control. For ground reaction is not implemented yet for this robot because there is no supporting mechanism or hardware.

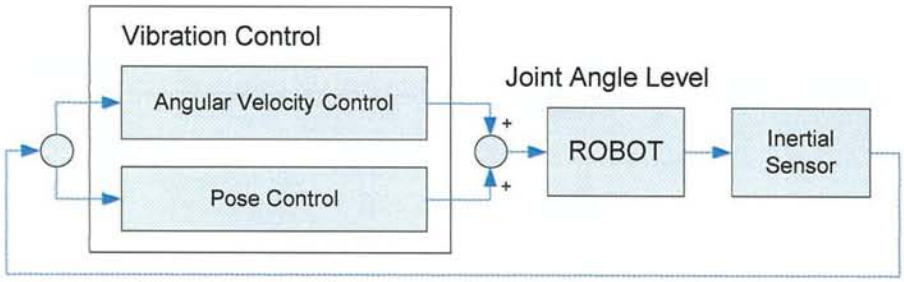



**Figure 4.26:** *Combining control diagram*

**4.3.2.2.2 Combination Control** To gain the balancing, robot must react when it get an influence, either internal or external influence, so the attitude behavior has been owned by the robot. EROS is implementing Pose Control and Angular Velocity Control.

To get the maximal stability level and reduce more vibration disturbance, we combine 2 internal stabilization in robot EROS, pose control and angular velocity control. We call them Vibration Control. We combine output of them on joint angle level, directly change angle of Servo Motor as shown in Fig. 4.26.

1) *Pose Control*: For this Pose Control, robot must maintain the bodys pose for some specific position. Robot holds the balancing condition that has been determined before. By maintaining the robots position in upright condition, so the robot condition will be more stable and it will be a basic of robots system stability to compensate the external influence.

Hand is used to compensate actuator for this control. It can be happened by controlling the arms angle towards the vertical line that influential to move the heavy-handed point towards heavybody point of this robot, so the gravity center will be in safe range. Approach center of mass attitude is to keep the center of gravity robot in the desired position in the area housed a foothold. By keeping the CoM remains in the footing area, the robot will be in a stable position. From Fig. 4.24., We can get some formulas below:

$$R = \frac{1}{M}\sum_{i=1}^{n} m_i r_i, R = 0 \tag{4.24}$$

$$\theta_3 = -\sin^{-1}\left(\frac{m_1 a_1 \sin(\theta_2) + m_2 a_2 \sin(\theta_1 + \theta_2)}{m_3 a_3}\right) \tag{4.25}$$

From this equation, we get $\theta_3$ which the angle between hand and vertical body. We can look at Fig. 4.24. This angle is the respond control for pose control.

For keeping the inclination body robot, pose control use PID controller. This controller, read the real time data from IMU sensor. The output of pose control is addition between output of center of mass and output of PID control as shown at Fig. 4.27. When robot in-

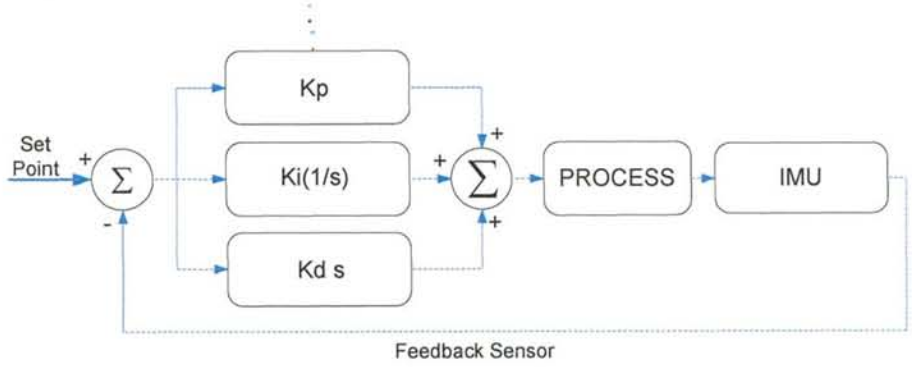



**Figure 4.27:** *PID Control Diagram of Pose Control*

cline forward, the robots hand will respond direction to backward and when robot incline backward, the robots hand will respond direction to forward.

2) *Angular Velocity Control*: The function of this control reduces the vibration disturbance which caused by motion vibration and ground reaction moment. To reach equilibrium, the magnitude of the torque on a system has to be zero. If the moment of Inertia has been set, then the Angular Velocity Parameter will determine the amount of torque. So, to get zero torque, the expected Angular velocity has to be zero. Joint actuator is used as a control output by changing the default angel of each joint in robots legs, so that it will not change the kinematics of motion in robots system.

$$\sum \tau = 0 \tag{4.26}$$

$$\tau_1 = \sum I.\omega \tag{4.27}$$

$$\tau_1 = \left(\frac{1}{2}m_1a_1 + \frac{1}{2}m_1(a_1 + a_2) + \frac{1}{2}m_1(a_1 + a_2 + a_3)\right)\omega \tag{4.28}$$

This control is expected to mitigate and reduce the impulse disturbance that resulted by some suddenly stoping and running. From Eq. (4.28), the response output from this control is the value of motor torque in ankle. The value of torque is controlled by PD controller. This control keeps the angular velocity which by gyroscope sensor stable in zero as shown at Fig.4.28.

**4.3.2.2.3 Motion modeling in proposed biped robot** Robot has been formed by some kind of movement, such as: forward movement, turn right, and turn left. Each movement has some level of speed from rest to a certain speed that has been set up by the long leg steps. To establish the movement speed of the robot, we set the timing period footsteps and set foot stride length. The faster the footsteps period, the greater the period of iteration steps starting

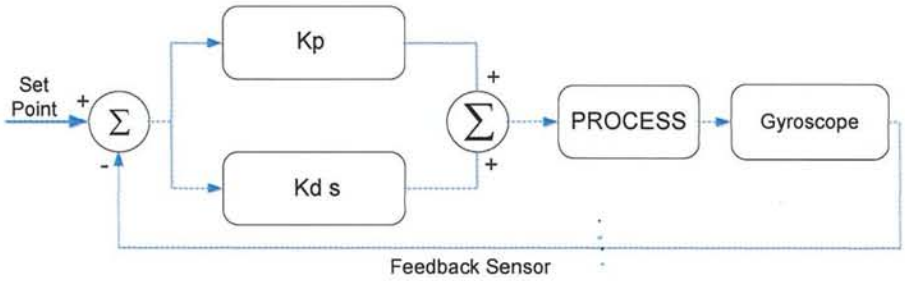



**Figure 4.28:** *PID Control Diagram of Angular Velocity Control*

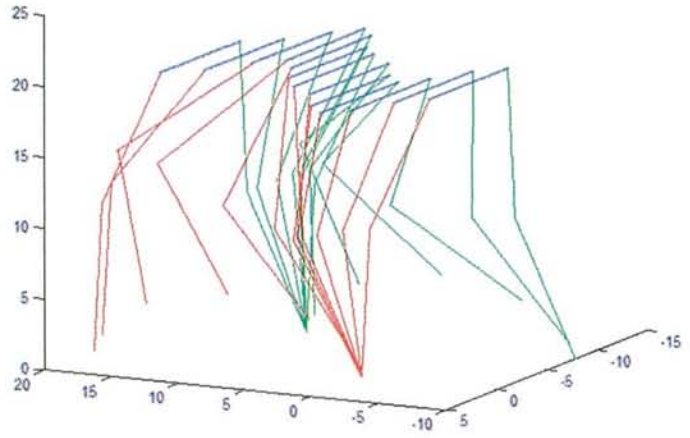



**Figure 4.29:** *Motion trajectory*

point to the end point measures. The model generates robot's movement step shown in Fig. 4.29.

In each sequence of movements, there are 2 conditions of DSP (Double Support Phase) and 2 conditions of SSP (Single Support Phase). At point center of mass in DSP conditions is required to be in a position between the two feet. While at CNS, the point of CoM is required to be at the central position of the soles of the feet.

Fig. 4.30 shows the step length changes when conditions SSP (Single Support Phase). When a step length A1 is given as an initial condition, at the time of the swing leg is at its peak, then the legs will be positioned to move the condition of A2.

When robot is activated, the angle join in robot has errors. This is caused backless gear in servo motors. So this error resulted in a change in the trajectory path. To reduce this local error in each join and to create a form of inverse kinematics in accordance with the desired, we use fuzzy algorithm. This fuzzy algorithm just keep the real angle joint suitable with the command from inverse kinematics system.

In this fuzzy control, we divided the error value in 7 membership function, which is shown in Fig. 4.31. The output from this control is the angle value which will be added in

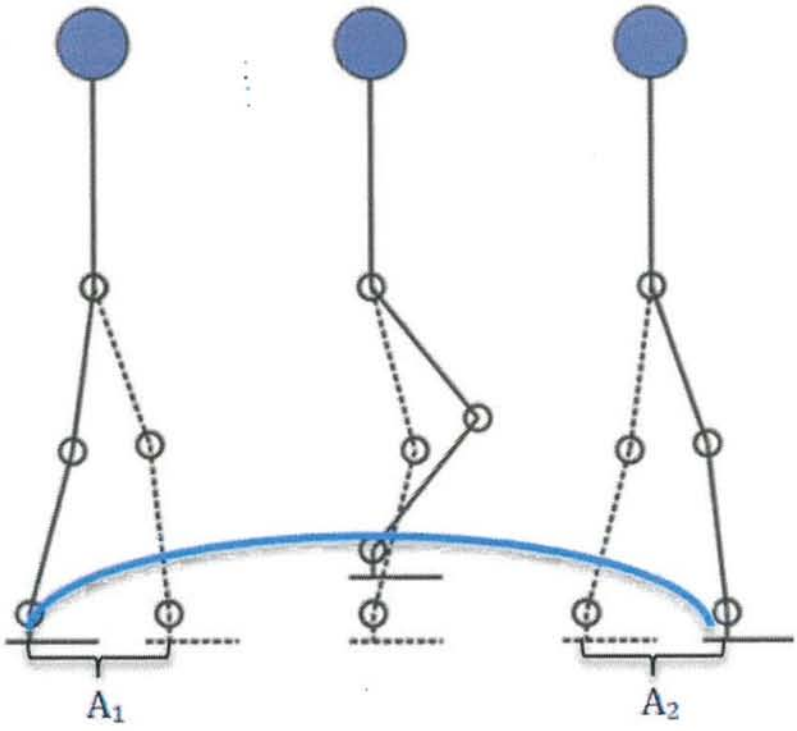



**Figure 4.30:** *Speed changing motion*

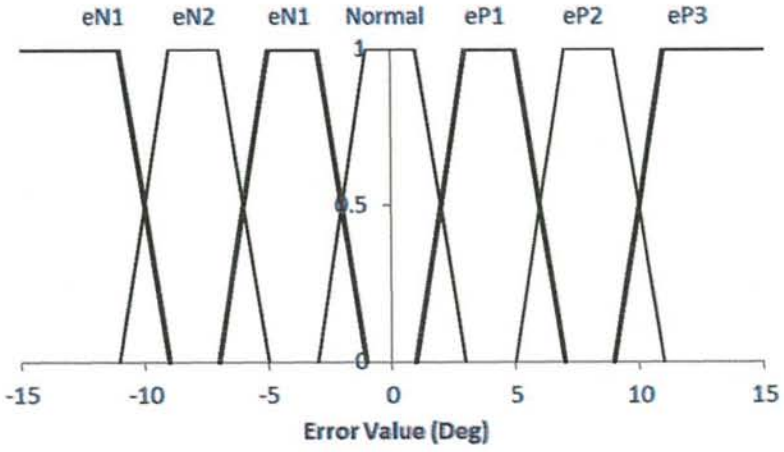



**Figure 4.31:** *Fuzzy membership functions*

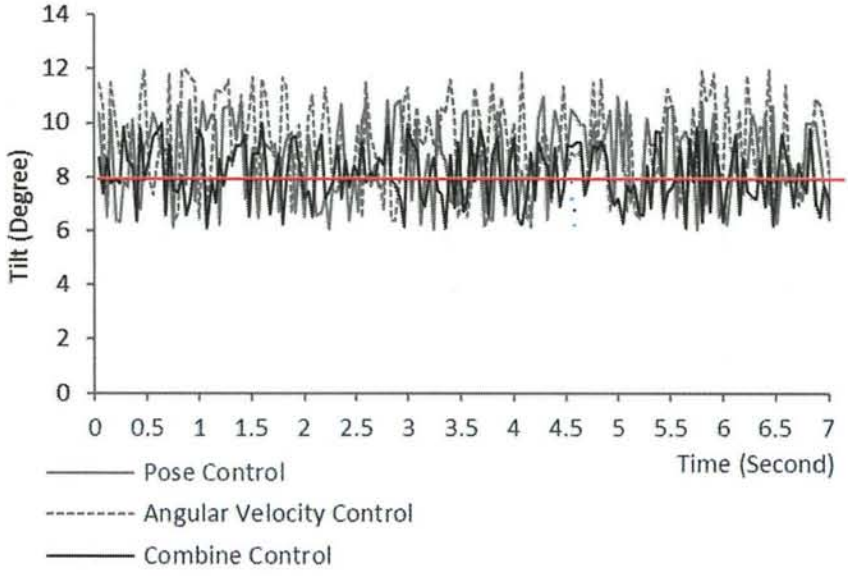



**Figure 4.32:** *Robot body's tilt oscillation Graphics*

the current angle join in servo motor. The angle value is got from defuzzyfication method. We make a simple system to compensate the backlash problem. So this system does not need many script program and does not disturb the main system.

#### 4.3.2.3 Experimental result

Scenario testing of this research is done by observing the tilt angle of the robot body with units of degrees resulting and angular velocity of the robot body from the movement of the robot in real time and single control condition and control condition combination. In this experiment, robot walks with normal speed (length of stride 50 mm). On this model calculation, we got the set point value of bodys robot angle, it is $8^o$. On this implementation, robot must being stable when it walks with appropriate angle by the set point.

By looking at the graph in Fig. 4.32, if we only use the controls pose, the robot has a robot body tilt oscillation rate of 5 degrees. Whereas if only using the angular velocity control, the robot has a robot body tilt oscillation rate of 6 degrees. By combining the control poses and angular velocity control, showed a more stable robot movement by having a low level of oscillation. The robot body tilt oscillation rate is 4 degrees.

Fig. 4.33 shows the angular velocity values taken from the gyroscope sensor located on the robot body. If we use only the pose control or using the angular velocity control, the oscillations of angular velocity value become larger, where its range is from -25 up to 24 for pose control and -19 up to 20 for angular velocity control. When we used combination control, we can reduce the oscillation of angular velocity. Its range is reduced from (-25 up

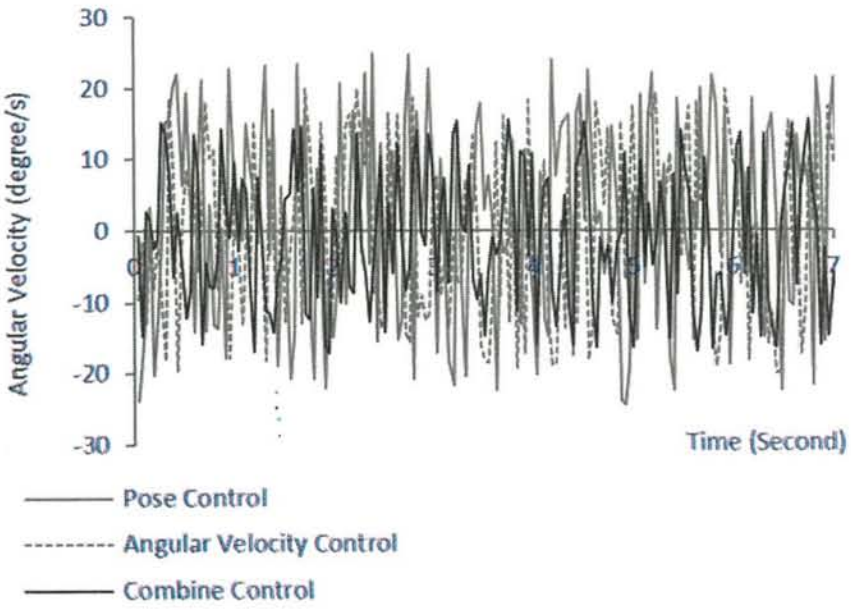



**Figure 4.33:** *Robot angular velocity oscillation Graphics*

to 24) to (-17 up to 16).

Then, by observing the movement of the robot in each of the speed or stride length with units of millimeters (mm) long that steps 0, 15, 30, 50, 70, 90, 105. We analyzed the performance of robot movement in each speed level as shown at Table 4.2.

On experiment of level speed (stride length), when using combinational control, robot capable to walk at higher speeds, at 105 mm length of stride. This speed level can not be implemented in only pose control or angular velocity control. Therefore, this experiment proves that combinational control effectively control the robot EROS (EEPIS Robot Soccer). Robot has stride ability as far as 105 mm. if robot steps by step length exceeding 105 mm, robot will fall. The control system can not support servo motor performance and foot length to cover that condition.

We also test the robot movement by some disturbance. When robot walk normally, we give force impulse to body of robot. We analyze the tilt of robot body. The result shows in Fig 4.34. The sampling time of the robot is 40 msec. On the 15th sampling time, the robot got a disruption and it caused the drastic changes on robot's tilt angle, and then robot stabilized itself with the control inside it, this needs 28 sampling time (1.12 seconds). To summarize, we conclude that this system is effective for increasing the degree of stablility level.

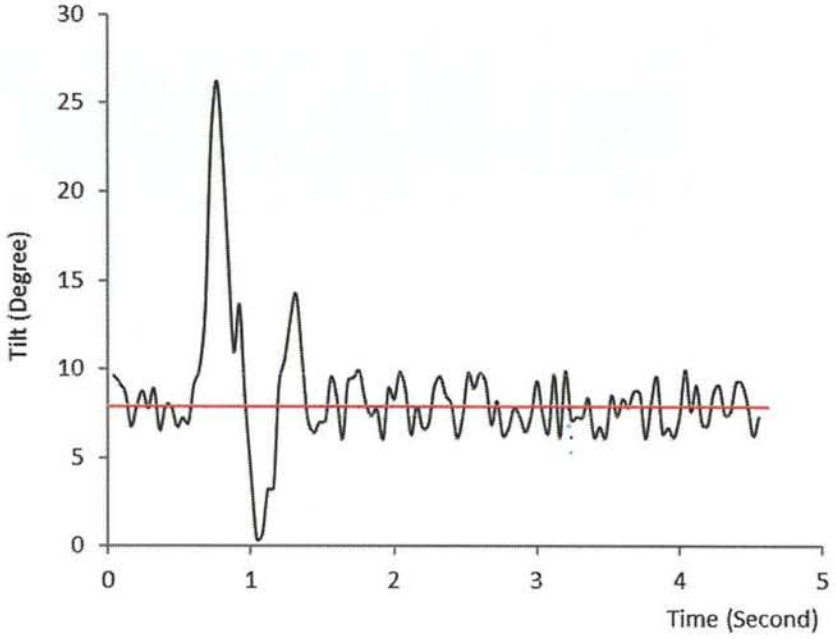



**Figure 4.34:** *Oscillation graphics when got disruption*

**Table 4.2:** *Walking Condition*

| Stride Length (mm) | Walking Condition | | |
|---|---|---|---|
| | *Pose Control* | *Angular Velocity Control* | *Combination Control* |
| 0 | Stable | Stable | Stable |
| 15 | Stable | Stable | Stable |
| 30 | Stable | Stable | Stable |
| 50 | Stable | Stable | Stable |
| 70 | Stable | Stable | Stable |
| 90 | Fall | Stable | Stable |
| 105 | Fall | Fall | Stable |

# Chapter 5

# Locomotion Generator Model

In this chapter, in order to avoid saturation development using conventional model, we proposed the new alternative way in locomotion generation model. we show new model inspired from neuroscience model, which is called central pattern generation model. The several learning models used for optimizing proposed locomotion model will also be shown. The aim of this research is for developing the new model of dynamic locomotion generator able to generate movement behaviors depending on the environmental condition. There are several model developed in this proposed research which have been modified and evoluted from previous model in order to achieve desired locomotion behavior generation. In the first development, we develop static interconnection structure of neural oscillator as the locomotion generator. After that, develop adaptive interconnection structure which able to change the structure depending on the environmental condition. In order to reduce the computational cost, we develop efficient neural structure. After that, we design walking speed controller in neural based locomotion generator, by changing the interconnection structure. Finally, we developed omni-directional walking controller using centered lerning strategy.

## 5.1 Neural Oscillator based Locomotion

The conventional approach does not represent well human behavior during locomotion. It also needs high mathematical complexity to realize the dynamic walking pattern. Therefore, we propose locomotion, based on a biological approach, that represents the human behavior system, and that can be stable and flexible, depending on the environmental condition. The controller system in the brain has been proposed by Roy [164]: this is the basis of how the brain controls locomotion. Locomotion models based on a biological approach have been proposed by several researchers [86, 84, 196, 136, 142]. Before we applied locomotion in

a biped humanoid robot based on the neuro-biological approach, we studied the locomotion system as adjusted by animal morphologies [51]. Four-legged animal locomotion has been proposed by Ijspeert et al. They control animal locomotion by using a neural oscillator and also design the transition mechanism from swimming to walking [86], [84], [85]. Locomotion based on the central pattern generation (CPG) approach in four-legged animal robots has been applied by several researchers [51]. Furthermore, CPG can also be applied for malfunction compensation in six-legged robots [158].

A neural oscillator is also implemented for biped robot locomotion, as proposed by several researchers [196, 136, 142, 88]. Taga et al. used coupled neurons for generating the oscillated signal to drive the joint. They dealt with a sensory feedback system to adapt to the environmental conditions and created a mathematical model in order to acquire the feedback calculation. Their proposed method is applied in computer simulation [196]. Another neuro-model of locomotion is presented by Matos et al., who proposed a CPG approach based on phase oscillators for bipedal locomotion [127]. However, the ability to recover the disturbance is required. Ishiguro et al. also proposed the concept of a neural oscillator to realize two-legged robot locomotion. This model is applied to control a three-dimensional biped robot that is intrinsically unstable. They applied a feedback sensor to form dynamic locomotion; however, the robot has a limited degree of freedom and is applied at simulation level only [88]. In 2014, Nassour et al. proposed the locomotion model in the humanoid biped robot: they extended the mathematical model of CPG and designed a multi-layered neuron connection in order to control various models of walking [142], [141]. Nassour's research has good stability; however, the aim of our research is to improve the stability level of walking. In 2010, Park et al. designed a locomotion using an evolutionary optimized CPG. They also proposed sensory feedback to support the walking model; however, they did not consider a learning system for stability [148]. Other researchers have considered center of mass (COM) for their locomotion, based on central pattern generation [79], [149]. They have not considered however the stability of the learning system, or control to get various walking patterns.

In this research, we use a neural oscillator adapted from the human mechanism to generate locomotion. However, a neural oscillator does not deal with variant movement control. We cannot control to get variant walking patterns (length of step and walking direction). Therefore, we modify and design a new inter-connection of four neurons: motoric neuron; sensoric neuron; command neuron; and gain neuron. In this research, the gain neuron receives the gain value from the command neuron that represents the energy for driving the joint movement. The neuron system is depicted in Figure 5.1.

One of our contributions to humanoid robot locomotion is that it is based on a biologi-

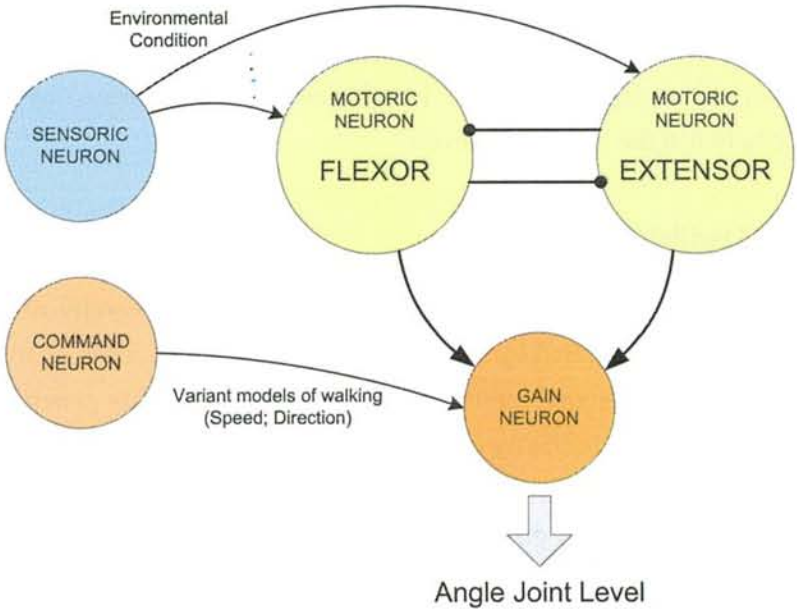



**Figure 5.1:** *Neuron model for locomotion*

cal approach. This research has advantages for generating variant motions having different lengths of movement and different walking directions.

Another novelty of our approach is that, while other researchers have used a single objective evolutionary algorithm for optimization, we have implemented a multi-objective evolutionary algorithm to determine the value of synapse weight between the motoric neurons. By using a multi-objective evolutionary algorithm, we can optimize two objectives at the same time: the walking speed and the stabilization for locomotion. Therefore, we can acquire the locomotion pattern with the maximum speed and with the best stabilization in its capability.

Another novelty of our research is in the stability learning system of locomotion, using a RNN to acquire the dynamic synapse weight between the motoric neuron and the sensoric neuron. The stability system learns the synapse weight between the sensoric neuron and the motoric neuron according to the feedback condition from the environment that can improve the robot's capability in stabilization. In order to prove the effectiveness of our system, we apply it in ODE simulation [183] and, in addition, we build a new biped robot.

### 5.1.1 Locomotion Model

In this section, we explain the architecture of the neural oscillator in the humanoid biped robot. First, we explain the structure in one joint angle. Then, the detailed inter-connected structure of the neuron will be explained. In one joint, there are three kinds of neuron: the motoric neuron that generates the rhythmic signal; the sensoric neuron that transmits the

feedback signal to the motoric neuron; and the command neuron that controls the gain of signal output from the coupled neuron. The neuron model is illustrated in Fig. 5.1.

In this section we emphasize how the locomotion pattern is formed by optimizing the weight synapse between the motoric neurons.

#### 5.1.1.1 Neural Oscillator

The basic neural oscillator has been proposed previously by several researchers [130, 129, 161]. This model is generated by mutual inhibition between certain neurons. Each neuron also acquires adaptation signal input. The rhythmic signal activity generated by the neural oscillator consists of two tonically excited neurons with the self-inhibition effect linked reciprocally via an inhibitory connection. Matsuoka discusses various aspects of frequency and pattern control: we can also generate different rhythm patterns by modifying parameters [130, 129]. Rowat et al. proposed a coupled neural oscillator with self-rhythmic generation ability. In this model, each neuron generates the signal to its pair. He divided the behavior pattern into six different types of neural oscillator [161].

Our neuron architecture model uses the neural oscillator proposed by Matsuoka as the basic element of the locomotion generator. Its mathematical formulation (which has been modified) is presented in Eq. (5.65), (5.66), (5.67).

#### 5.1.1.2 Inter-connection Model

The inter-connection of multi-neurons influences the pattern of the rhythmic signal. Lakany et al. previously presented and analyzed the walking pattern in humans by using 3D motion equipment based on infrared reflective markers. They showed the joint trajectory of hip, knee, and ankle in human walking [107]. We designed the inter-connection model by analyzing the human walking pattern. Matsuoka presented several combined inter-connections of multi-neurons in a neural oscillator [130, 129]. Some inter-connections are suitable in human locomotion. The neuro-connection in our model has four motoric neurons for the main pattern to generate the rhythmic signal. Other neurons follow the main rhythmic signal. In our model, depicted in Fig. 5.2, the motoric neuron receives two different types of sensory information: proprioceptive and exteroceptive information. The external input formula is extended to support the dynamical oscillator.

In order to realize multi-sensor integration, we design the sensoric neurons as the feedback sensor connected to the certain motoric neuron. The weight values that represent the effects of the feedback sensor to each motoric neuron are determined by using the RNN model. We have three weights in the sensoric neurons: $\mathbf{Y}^{(1)}$ represents the influence of the

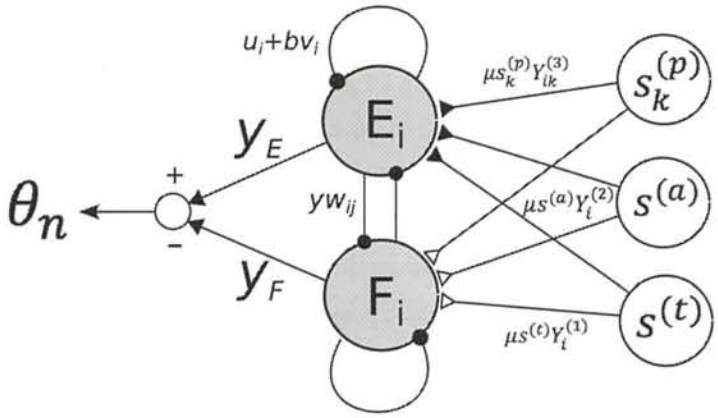



**Figure 5.2:** *Neural oscillator diagram*

inclination sensor; $\mathbf{Y}^{(2)}$ represents the influence of the angular velocity sensor; $\mathbf{Y}^{(3)}$ is the weight value representing the influence of the ground sensor to each neuron placed on the sole of each foot. These weight parameters result from the stability learning system and are optimized in real time. If there are some impulses from the feedback sensor, then the neural oscillator computed in Eq. (5.26), (5.65), (5.66), (5.67) receives the influence signal. Therefore, the joint angle generated by the neural oscillator is changed. In Fig. 5.2, $\mu$ represents the negative or positive signal, depending on the connection information in Table 6.3. A detailed explanation of the sensor connection will be provided in the next subsection.

$$D_s = \sum_{k=1}^{m} Y_{ik}^{(3)} s_k^{(p)} + Y_i^{(2)} s^{(a)} + Y_i^{(1)} s^{(t)} \tag{5.1}$$

$$\tau \dot{u}_i = (u_0 - u_i - \sum_{j=1}^{n} w_{ij} y_j - b v_i - D_s)\tau_f \tag{5.2}$$

$$\tau' \dot{v}_i = (y_i - v_i)\tau_f \tag{5.3}$$

$$y_i = \max(u_i, 0) \tag{5.4}$$

where $u_i$, $y_i$, and $v_i$ are the inner state, output value, and a variable that represents the adaptation value or the self-inhibition effect of the $i$th neuron; $b$ is the rate of the adaptation value. The external input for coupled neurons, which has a constant rate, is denoted by $u_0$. The time constant of the inner state and the adaptation effect in the neuron are represented by $\tau$ and $\tau'$, respectively. In Eq. (5.65), $w_{ij}$ represents the strength of the inhibitory effect between the motoric neurons that is optimized offline; $\sum_{j=1}^{n} w_{ij} y_j$ represents the total of the signal input from the neuron. In Eq. (5.65) and (5.66), $\tau_f$ is used for controlling the frequency of oscillation. By manipulating the inhibitory effect, the inner state time, the external effect, and the

inter-connection structure, we can produce and generate different rhythmic patterns.

In Eq. (5.26), $D_s$ is the feedback value from the sensoric neuron; $s_k^{(p)}$, $s^{(a)}$, and $s^{(t)}$ represent the output value from the touch ground sensoric neuron in node $k$, from the angular velocity sensoric neuron, and from the tilt sensoric neuron, respectively. Notation $m$ represents the number of touch sensors. In this research, we have eight touch sensors: four nodes on the sole of each foot. In Eq. (5.5), the angle of the joint is formed by mutual inhibition of two neurons, $y_{2n}$ and $y_{2n+1}$, influenced by the signals $\mu_{dir}$ and $\mu_{gain}$ from the command neuron computed in Eq. (5.6), where $\mu_{dir}$ and $\mu_{gain}$ are parameters for the direction that influences the hip-z joint ($n$ =11; 12) and the length of step that influences the hip-x, knee, and ankle-x joints ($n$ =1; 2; 3; 4; 7; 8), respectively. In order to control the locomotion pattern on a sloping surface, we control the parameter $\vartheta_{slope}$, which is a special parameter for the angle of ankle-y computed in Eq. (5.7).

$$\Theta_n = (y_{2n} - y_{2n+1})\, H_n + Q_n \tag{5.5}$$

$$H_n = \begin{cases} \mu_{gain} & \text{if } n = \{1, 2, 3, 4, 7, 8\} \\ \mu_{dir} & \text{if } n = \{11, 12\} \\ 1 & \text{otherwise} \end{cases} \tag{5.6}$$

$$Q_n = \begin{cases} \vartheta_{slope} & \text{if } n = \{9, 10\} \\ 0 & \text{otherwise} \end{cases} \tag{5.7}$$

Equations (5.65) and (5.66) were calculated using the Runge-Kutta-Gill method [200]. As seen in Eq. (5.67), the inner state outputs were defined as positive numbers. Two coupled neurons (flexor and extensor) represent one joint system. The robot in this model, with two legs and two hands, has 16 degrees of freedom (DoF): each leg has six joints and each hand has two joints. There are three joints in the hip position: the hip-x joint is rotational in the x-axis; the hip-y joint is rotational in the y-axis; and the hip-z joint is rotational in the z-axis. There is one joint at the knee. Furthermore, there are two joints at the ankle: the ankle-x joint is rotational in the x-axis and the ankle-y joint is rotational in the y-axis.

$$\theta_n^1 = \begin{cases} \Theta_n & \text{if } -\pi < \Theta_n < 0 \\ -\pi & \text{if } \Theta_n \leq -\pi \\ 0 & \text{otherwise} \end{cases} \tag{5.8}$$

$$\theta_n^2 = \begin{cases} \Theta_n & \text{if } -\pi/2 < \Theta_n < \pi \\ -\pi/2 & \text{if } \Theta_n \leq -\pi/2 \\ \pi/2 & \text{otherwise} \end{cases} \tag{5.9}$$

$$\theta_n(t) = \begin{cases} \theta_n^1(t) & \text{if joint knee} \\ \theta_n^2(t) & \text{otherwise} \end{cases} \tag{5.10}$$

There are some limitations in each joint, computed by Eqs. (5.42) ,(5.43), (5.68), where $n$ is the joint number and $\theta_n^2$ is the notation for the knee joint. In the inter-connection between the motoric neurons, there are server neurons and client neurons. The server neurons $E_1$, $E_2$, $E_3$, $E_4$ generate the main signal to the client neurons. The server neurons are located at hip-x and at the knee joint. In order to optimize and to summarize the weights of the synapses, we divided the weight synapse groups into seven clusters depending on the muscular system of the human leg. Those clusters represent the genes in the evolutionary process, as shown in Table 5.1. The weights in the same cluster have the same value. The detailed inter-connection model of the motoric neurons is illustrated in Fig. 5.3.

**Table 5.1:** *Clusters of Weights in the Optimization*

| Cluster | Weight connection parameter |
|---|---|
| $S_1$ | $w_{i,j}$ where $\{i;j\}$ : $\{1;2\},\{2;1\},\{3;4\},\{4;3\},\{5;6\},\{6;5\},\{7;8\},\{8;7\},$ $\{9;10\},\{10;9\},\{11;12\},\{12;11\},\{13;14\},\{14;13\},$ $\{15;16\},\{16;15\},\{17;18\},\{18;17\},\{19;20\},\{20;19\},$ $\{21;22\},\{22;21\},\{23;24\},\{24;23\},\{25;26\},\{26;25\},$ $\{27;28\},\{28;27\},\{29;30\},\{30;29\},\{31;32\}.$ |
| $S_2$ | $w_{i,j}$ where $\{i;j\}$ : $\{1;3\},\{3;5\},\{5;7\},\{7;1\}$ |
| $S_3$ | $w_{i,j}$ where $\{i;j\}$ : $\{1;3\},\{5;7\}$ |
| $S_4$ | $w_{i,j}$ where $\{i;j\}$ : $\{2;14\},\{1;13\},\{4;14\},\{3;13\},\{5;15\},$ $\{6;16\},\{7;15\},\{8;16\}$ |
| $S_5$ | $w_{i,j}$ where $\{i;j\}$ : $\{1;9\},\{1;18\},\{1;11\},\{1;20\},\{3;9\},\{3;11\},\{3;18\},$ $\{3;20\},\{5;10\},\{5;17\},\{5;12\},\{5;19\},\{7;10\},\{7;12\},$ $\{7;17\},\{7;19\}$ |
| $S_6$ | $w_{i,j}$ where $\{i;j\}$ : $\{1;28\},\{2;27\},\{5;32\},\{6;31\}$ |
| $S_7$ | $w_{i,j}$ where $\{i;j\}$ : $\{9;26\},\{10;25\},\{11;30\},\{12;29\}$ |

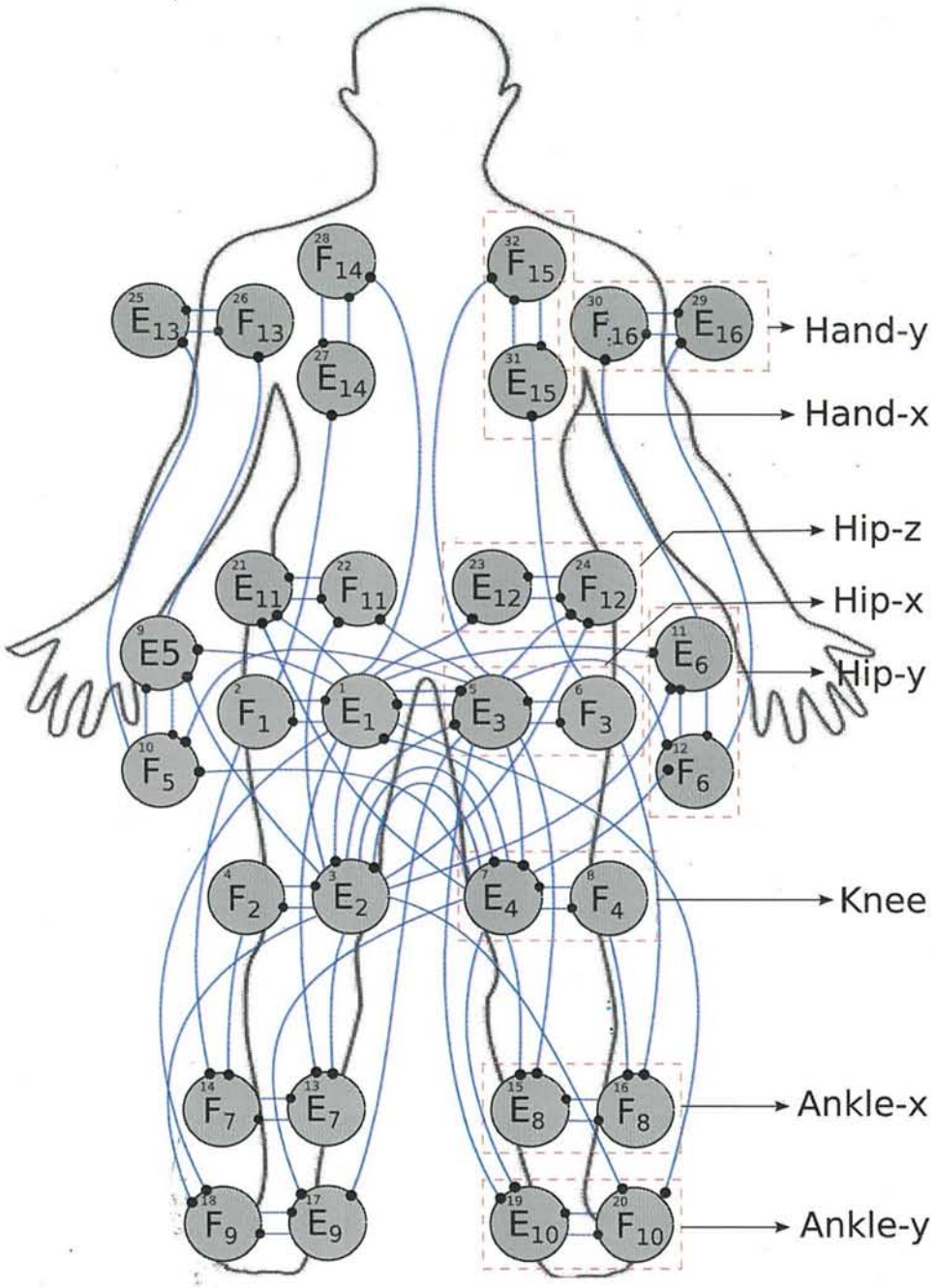



**Figure 5.3:** *Inter-connection model of the motoric neurons*

#### 5.1.1.3 Synapse Optimization

In this section, we will explain the optimization algorithm to acquire synapse weights for the locomotion pattern. Synapse weights represent the strength of the neurotransmitter that transmits the electrical signal from one neuron to the other neurons. There are many methods to optimize the synapse weights in the inter-connected system of neural oscillators. Many researchers have applied an evolutionary algorithm for optimizing their locomotion system based on a neural oscillator [79, 149, 22, 87, 31, 157]. An evolutionary algorithm is used by Baydin to determine the value of synapse weights and locomotion parameters: he ignored the stability effect in his model and realized 2-D biped robot locomotion [22]. Inada et al. also applied an evolutionary algorithm in searching for the parameter of central pattern generation in their bipedal robot locomotion model [87]. Hong et al. applied two-phase evolutionary programming [79, 149]. Chernova et al. used an evolutionary algorithm for gait learning in a four-legged robot [31].

In our research, we use an evolutionary algorithm to find the best value of synapse weights that represent the value of effect in the inter-connection between the motoric neurons. One individual contains seven genes representing the seven weight values, from $S_1$ to $S_7$ in Table 5.1. The weight values presented in Table 5.1 correspond to Fig. 5.3; therefore, the weights in Table 5.1 are the weights between the neurons in Fig. 5.3. Each neuron in Fig. 5.3 has an ID: Table 5.1 should be understood based on these IDs. For example, the weight between neuron 2 and neuron 27 is the weight between the $F_1$ and $E_{14}$ neurons in Fig. 5.3. This weight belongs to the group $S_6$. The weights in one group are equal to each other: thus, as we have seven groups, there are seven different weight values. This inter-connection model is created empirically with basic inter-connection model analysis [130, 129] and some preliminary tests.

#### 5.1.1.4 Single Objective Evolutionary Algorithm

In this research, the optimization process was conducted to find the value of effect between neurons. There are many methods for parameters optimization. In this research, multi-objective optimization is used since we have three objective optimization problems. For solving multi-objective optimization problems, there are two main methods: Pareto front and weighting factor. One of the main drawbacks of these methods is choosing the most appropriate value of weighting factors. The development of locomotion based on neural oscillator implies the optimization of the oscillation of body tilt $\bar{\sigma}$ (minimization) in pitch and roll direction, the change rate of walking direction (minimization) for maintaining the go straight movement, and the velocity of movement $\bar{v}$ (maximization). In order to acquire the mini-

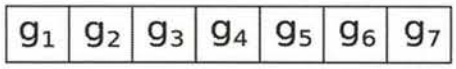



**Figure 5.4:** *Chromosome of an individual*

mization value represent by $\bar{v}$, we used the remaining parameter of velocity, which can not be reached by robot walking in certain time. In the optimization case, we calculated the absolute average value of body tilt in pitch and roll direction, which was computed using Eqs. (6.1), (5.33) and the velocity of robot walking was computed using Eq. (5.15). The weight among neuron should be determined to acquire the desired velocity with minimum oscillation of body tilt. In order to acquire the fitness evaluation computed by Eq. (5.56), we run the robot in ODE and analyze the output of sensor.

$$\sigma(t, w_{ij}) : |\mathrm{R}| \tag{5.11}$$

$$\mathrm{L}(t, w_{ij}), \mathrm{x}(t), \mathrm{y}(t) : \mathrm{R} \tag{5.12}$$

In Equations (5.31), the body tilt ($\sigma$) is normalized in absolute value. And in Eq. (5.32), L, x, y are the real number represented the length of movement in both x-axis, and y-axis respectively. These parameters were acquired in a time sampling during simulation process in ODE.

$$\bar{\sigma}_{pitch}(w_{ij}) = \left({}^{1}/_{T}\right) \sum_{t=0}^{T} \sigma_{pitch}(t, w_{ij}) \tag{5.13}$$

$$\bar{\sigma}_{roll}(w_{ij}) = \left({}^{1}/_{T}\right) \sum_{t=0}^{T} \sigma_{roll}(t, w_{ij}) \tag{5.14}$$

$$\bar{v}(w_{ij}) = \left({}^{1}/_{T}\right) \sum_{t=0}^{T} \frac{\mathrm{d}}{\mathrm{dt}} \mathrm{L}\big(t, w_{ij}, \mathrm{x}(t), \mathrm{y}(t)\big) \tag{5.15}$$

$$fa = \arg\min_{w_{ij}} a(\bar{\sigma}_{pitch}\varepsilon_1 + \bar{\sigma}_{roll}\varepsilon_2 + \bar{v}\varepsilon_3) \tag{5.16}$$

We implemented steady state genetic algorithm (SSGA) for optimizing the value of weight among neuron ($w_{ij}$). In SSGA, there is one individual that inserted to the new population in one generation. We used weighting factor in order to optimize multi-objective problems. $\varepsilon_1$, $\varepsilon_2$, $\varepsilon_3$ are the weight coefficients for tilt oscillation in pitch direction, roll direction, and for walking velocity respectively. In SSGA process, first we select the parent from initial generation; second we create the new individual by using mutation and crossover process. After that we evaluate the new individual by using fitness calculation computed in Eq. (5.56).

In SSGA, the chromosome of the individual is composed of the weight parameters among neuron. Each parameter in represented by one gene. Since we have seven parts of weights,

thus one individual has seven genes represented in Fig. 5.4. Each gene has minimum ($h_{min}$) and maximum value ($h_{max}$). The detail of mutation and crossover process was explained in our previous research [177].

#### 5.1.1.5 Multi-objective Evolutionary Algorithm

While current models have used a single objective evolutionary algorithm in neuro-locomotion, we use a multi-objective genetic algorithm in relation to two objectives.

$$St(t, w_{ij}) : |\mathrm{R}| \tag{5.17}$$

$$\ell(t, w_{ij}); \mathrm{x}(t); \mathrm{y}(t) : \mathrm{R} \tag{5.18}$$

Our aim is to minimize the fitness functions at the same time: minimization of tilt body $\bar{St}$ and minimization of the remaining distance from target of walking length $\bar{v}$. The value of tilt body oscillation represents the stability of movement. If the robot locomotion has low oscillation, it implies good stabilization. Our goal is to realize a locomotion pattern that has good stabilization. The remaining distance represents the speed of the robot walking. If the robot has a high value in the remaining distance, the robot has a low speed in walking. Our other goal is to realize a locomotion pattern with maximum possible speed. Therefore, our objectives are to acquire locomotion that has good stabilization and a high walking speed.

In Eq. (5.17), $St(t, w_{ij})$ is tilt oscillation that has absolute value. In Eq. (5.18), $\ell(t, w_{ij})$ is the resultant value of $\mathrm{x}(t)$ and $\mathrm{y}(t)$ in each time sampling. The $\ell(t, w_{ij})$, $\mathrm{x}(t)$, $\mathrm{y}(t)$ notations were defined as real numbers.

$$\bar{St}_{(w_{ij})} = \frac{1}{T} \sum_{t=0}^{T} St(t, w_{ij}) \tag{5.19}$$

$$\bar{v}_{(w_{ij})} = \Upsilon - \frac{1}{T} \sum_{t=0}^{T} \frac{\delta}{\delta t} \ell\Big(t, w_{ij}, \mathrm{x}(t), \mathrm{y}(t)\Big) \tag{5.20}$$

$$w_{ij}a = \arg\min_{w_{ij}} a\bar{St} \tag{5.21}$$

$$w_{ij}a = \arg\min_{w_{ij}} a\bar{v} \tag{5.22}$$

The fitness is computed by Eqs. (5.19), (5.34), (5.56), (5.57). In Eq. (5.34), $\Upsilon$ is the desired length of the robot movement and $T$ is the maximum time sampling. All sensory data in the optimization are based on the data resulting from ODE computer simulation.

In the multi-objective evolutionary algorithm, we need fast non-dominated sorting and good spread of individuals in the Pareto front. Therefore, we apply the NSGA-II algorithm proposed by Deb et al. [36] for optimizing the robot locomotion. Several researchers have used NSGA-II for high computational problems. This evolutionary algorithm applies fast non-dominated sorting and diversity preservation.

In fast non-dominated sorting, the solutions within one population are compared in order to acquire the first non-dominated front. For one generation, this sorting algorithm requires $O(MN^2)$ computational complexity, where $N$ is the population size and $M$ is the number of objectives. In the case of multi-objective algorithms, the optimal solutions (the solutions in the so-called Pareto front) consider more than one objective. Sometimes, solution 'A' is better than solution 'B' in one objective function, but in other objective function solution 'B' is better than solution 'A'. We may acquire more than one solution in multi-objective problems.

In diversity preservation, the goal is to acquire a good spread of the solutions in the Pareto front. Our aim is to have good diversity of the solutions in order to acquire variant patterns of walking. In NSGA-II, the density and crowding of solutions in the Pareto front is considered for diversity preservation: these are described by density estimation and the crowded comparison operator. In order to acquire density of solutions, average distance calculation of two points is required. The computational complexity of this method is $O(M \cdot N \cdot \log N)$, where $M$ is the number of independent sortings and $N$ is the number of solutions. The selection process for individuals is based on the crowded comparison operator [36].

Some researchers have successfully applied NSGA-II for multi-objective problems [138], [110], [90]. In our problem, we evaluate two objectives, based on the sensory feedback from the computer simulation explained above. The details of the parameters will be explained in Section 5.1.3.

### 5.1.2 Experimental Result using Single Objective Model

We designed the robot by using computer simulation ODE adapting the rules of RoboCup 2015. Parameter properties from EROS robot were inserted in simulation robot as shown in Table 5.2. We conducted the experiment gradually as follow; first, optimizing the weight among neuron ($w_{ij}$) to acquire the walking locomotion pattern; second, optimization for increasing the stability in locomotion. In order to find the pattern of trajectory, we observed the signals generated from many combinations of interconnection in server neuron introduced by Matsuoka. After the best pattern of locomotion was determined, we optimize the synapse weight among neurons. SSGA algorithm was applied that use speed of walking ($\bar{v}$)

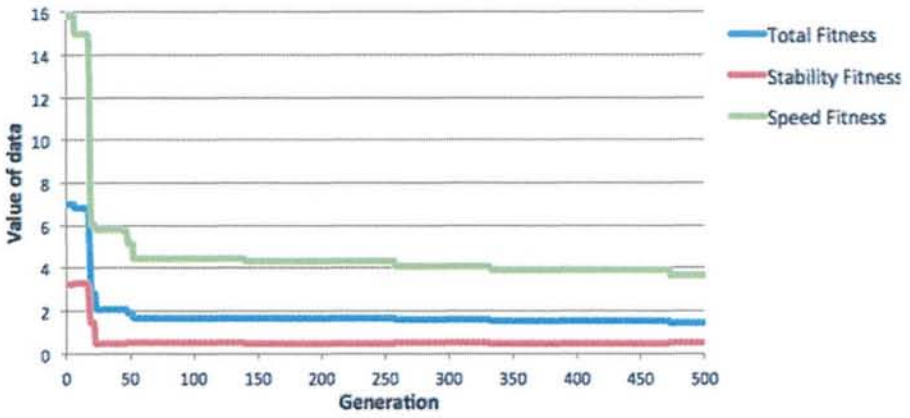



**Figure 5.5:** *Fitness evolution*

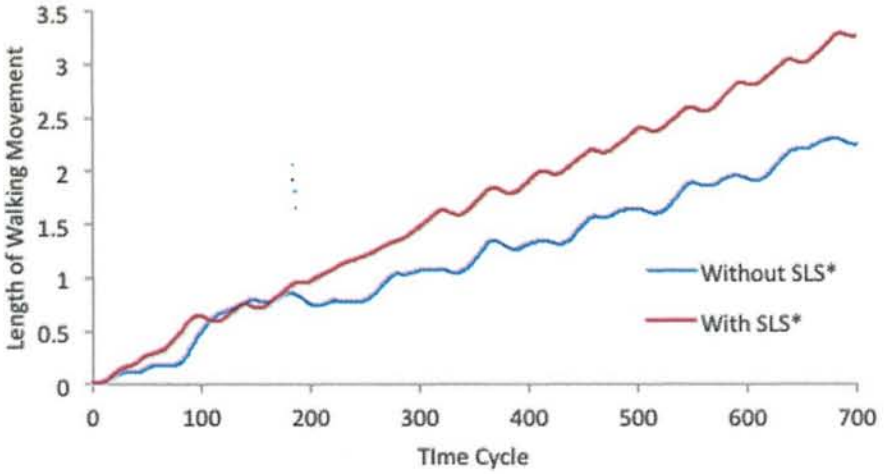



**Figure 5.6:** *Length of walking comparison*

and tilt sensor ($\bar{\sigma}$) as the objective function. The parameters used for weight among neuron optimization was tabulated in Table 5.3.

In this experiment, we acquire the fitness diagram from SSGA process that shown in Fig. 5.5 resulted from two objective values (Speed and Stabilization). The fitness total decreased significantly until 50-th generation. Stability fitness and speed fitness also decreased, which implied the oscillation data in pitch and roll direction to become smaller and the speed become higher. The good pattern locomotion were reach in 50-th generation. We finished the generation until 500-th generation. The locomotion system takes one individual (S1-S7 = 1.546, 1.230, 1.901, 1,676, 2.158, 1.390, 1.120) as the best individual in SSGA, and then become the parameter in robot. Next, we analyzed the signal resulted from coupled motoric neuron.

By using evolutionary algorithm, walking pattern based on neural oscillator can be

**Table 5.2:** *Robot specification*

| Parameter | Description |
|---|---|
| Degree of freedom | 18 degree of freedom |
| Height, weight | 600 mm, 3000 gram |
| Sensors | 4 ground detection sensors in each leg<br>Tilt sensor and angular velocity sensor |

**Table 5.3:** *Parameters of SSGA*

| Parameter | value |
|---|---|
| Population size | 16 individuals |
| Num. of generations | 500 generations |
| Num. of Objectives | 2 objectives: speed and stabilization |
| Chromosome | 7 real number |
| Evaluation time | 9 second |
| $\varepsilon_1, \varepsilon_2, \varepsilon_3$ | 0.3, 0.3, 0.7 |

formed. However, since the stability level resulted from evolutionary computation was not strong enough to cover outside disturbances, the stability system is required to solve this problem.

### 5.1.3 Experimental Result using Multi Objective Model

This section presents the experimental results of the proposed model. Here, the experiments are divided into two parts: simulation conducted by ODE and the real robot experiment.

In order to show the effectiveness of the proposed model, several types of experiment were conducted on the real robot: different steps for walking, different directions for walking, and on a sloping surface. The neural oscillator's parameters $\tau$; $\tau'$; $\tau_f$ were set as 12.0; 1.2; 1.0 respectively, and we also set $b$ as 2.5 and $u_0$ as 1. In this experiment, the time cycle is 0.01 seconds.

#### 5.1.3.1 Simulation Experiment

In the simulation, we constructed the simulation robot system based on the real robot properties explained in Table 5.4. At the beginning of the experiment, the weight synapse was optimized using a multi-objective evolutionary algorithm. The acquired optimum weight

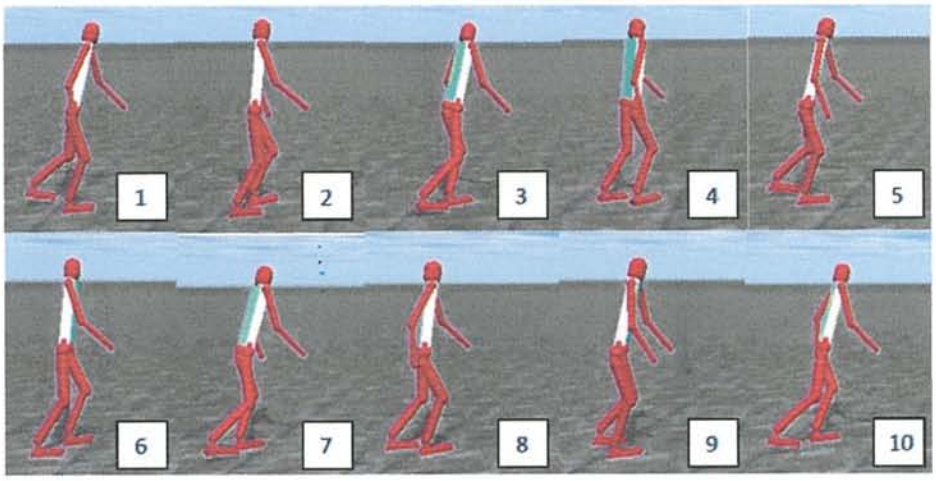



**Figure 5.7:** *The robot walking in simulation*

parameters were used in the next simulation experiment, as well as in the real robot experiments. Furthermore, we implemented the stability system in the robot simulation. In addition, we conducted analysis of the relationship between the output of the gain neuron, the feedback from the sensoric neuron, and the timing of the locomotion.

**Table 5.4:** *Real Robot Properties Corresponding to the Robot Simulation*

| Part | Length | Weight |
|---|---|---|
| Leg 1 (from knee to ankle) | 98 mm | 0.25 kg |
| Leg 2 (from hip to ankle) | 98 mm | 0.2 kg |
| Body | 130 mm | 0.6 kg |
| Top elbow | 85 mm | 0.18 kg |
| Bottom elbow | 85 mm | 0.18 kg |

**5.1.3.1.1 Experiment for optimizing weight synapse** In order to acquire the best locomotion pattern based on neural oscillation, we optimized the weight between the motoric neurons by using a multi-objective evolutionary algorithm. The parameters used in this method can be seen in Table 5.15, while the gain parameter ($\mu_{gain}$) was defined as 1.0 and the direction parameter ($\mu_{dir}$) was defined as 0. The applied parameter setting provided us with a good solution. Figure 5.45 shows the simulation process when the robot was walking.

For the optimization method, we ran the NSGA-II with 32 individuals and with 64 individuals. The training process took nine hours for 32 individuals and 17 hours and 50 minutes for 64 individuals. However, the evolutionary operations took only 0.832 seconds and 1.17

**Table 5.5:** *NSGA-II Parameters*

| Parameter | Exp. 1, Exp. 2 |
|---|---|
| Population size | 32 indv., and 64 indv. |
| Number of generations | 100 gen., 100 gen. |
| Number of objectives | 2 obj. Speed and Stabilization: |
| Chromosome | 7 real numbers |
| Crossover & Mutation prob. | 0.3, 0.3 |
| Time evaluation | 10 second |

seconds, because the training time was spent much more on the evaluation process, which applies real time simulation in ODE. As the result, we acquired the Pareto front as depicted in Fig. 5.8. We may choose more than one solution from the Pareto front. We choose the solution that has a good compromise between the first objective and the second objective. Therefore, we can acquire the parameters related to fast walking with good stabilization.

From the Pareto front produced by the optimization process, four solutions were chosen as the result of the evolutionary algorithm process. Therefore, we have four different combinations of synapse weights in neural locomotion: $S_1 - S_7 = \{1.246, 1.703, 1.415, 0.762, 1.117, 2.436, 1.540\}$ as the first solution; $S_1 - S_7 = \{1.246, 1.703, 1.415, 0.762, 1.117, 2.436, 1.540\}$ as the second solution: $S_1 - S_7 = \{1.246, 1.703, 1.415, 0.762, 1.117, 2.436, 1.440\}$ as the third solution; and $S_1 - S_7 = \{1.534, 1.430, 1.415, 1.362, 1.717, 1.436, 1.670\}$ as the fourth solution. We ran the robot using these parameters, and analyzed the data oscillation (angular velocity data and tilt angle oscillation). The oscillation data of the first, second, and fourth solutions are better than those of the third solution. The third solution had higher oscillation than all other parameter solutions.

This multi-objective computation can solve the two problems of walking speed and stability in one optimization process. In this system, for reducing the computational time, the clustering of weight parameters was used to reduce the number of genes in the individuals. Comparing to previous research, Baydin [22] covers only five degrees of freedom, while our system can cover 16 degrees of freedom of the humanoid robot. Not only can the evolutionary computation in our robot form the walking pattern, but it can also consider the stability of the robot. However, since the stability level resulted from evolutionary computation, it is not strong enough to support a long step; thus, the stability learning system is required to solve this problem.

In conclusion, this experiment shows the advantages of using a multi-objective evolu-

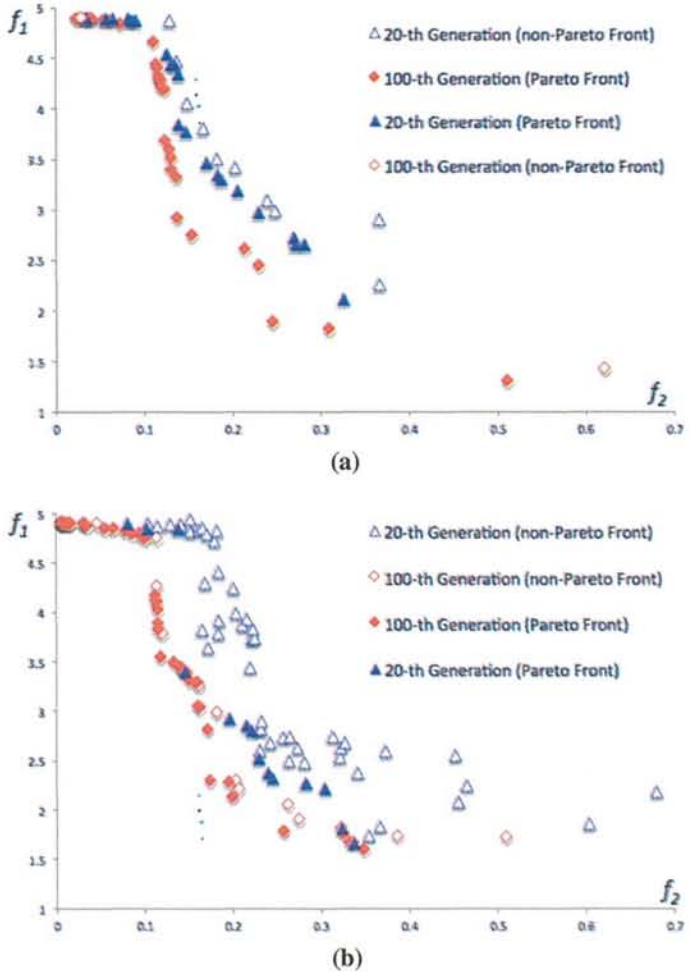



**Figure 5.8:** *a) Pareto front with 32 individuals; b) Pareto front with 64 individuals*

tionary algorithm to acquire the best speed in robot capability and minimal oscillation.

**5.1.3.1.2 Experiment for variant locomotion** We conducted this experiment in order to show the effectiveness of the locomotion with different lengths of step and different directions of walking. In order to acquire the variant length of step, we manipulated the gain parameter ($\mu_{gain}$) from 0.6 to 1.2. The output result for the neural oscillator in this experiment can be seen in Fig. 5.9. We ran the robot with Gain = 0.6 in time sampling 1-400, Gain = 0.7 in time sampling 401-800, Gain = 0.8 in time sampling 801-1200, Gain = 0.9 in time sampling 1201-1600, Gain = 1.0 in time sampling 1601-2000, and Gain = 0.9 in time sampling 2001- 2400. The tracking movement of the robot torso in this experiment is shown in a two-dimensional map in Fig. 5.10.

In the other experiment related to variant locomotion, we manipulated the direction parameter ($\mu_{dir}$) that represents the direction of walking. We set the direction parameter 0.09 in time sampling 1–800, -0.09 in time sampling 801–1600, 0.15 in time sampling 1601–2400, and -0.15 in time sampling 2401–3200. The result of this experiment is depicted in Fig. 5.11, showing the changing signal of the neural oscillator in hip-z joint. We also tracked the walking movement in a two-dimensional space (Fig. 5.12). When the direction parameter value increased, then the degree of turning also increased.

In conclusion, this experiment proves that biped robot locomotion can be manipulated into variant direction and speed of walking. Other researchers have not considered controlling variant walking speed and direction.

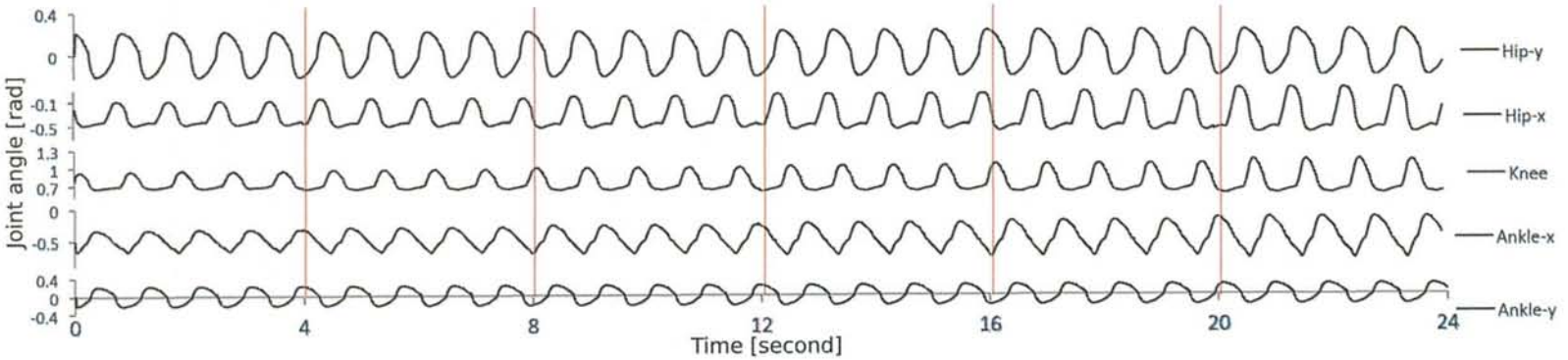



**Figure 5.9:** *The changing output of neural oscillator of robot while walking, using different gain parameters in certain time sampling.*

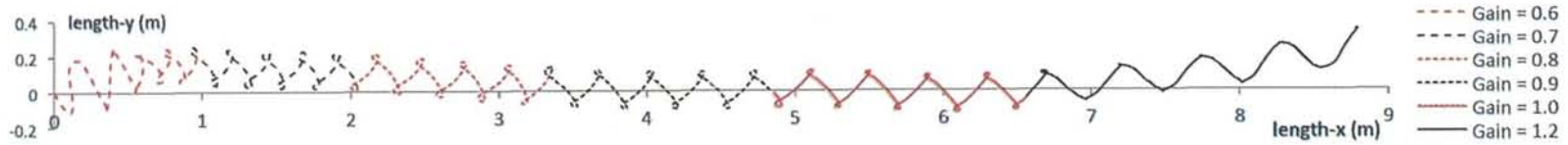



**Figure 5.10:** *Two-dimensional tracking from the center of mass in robot with different gain parameters*

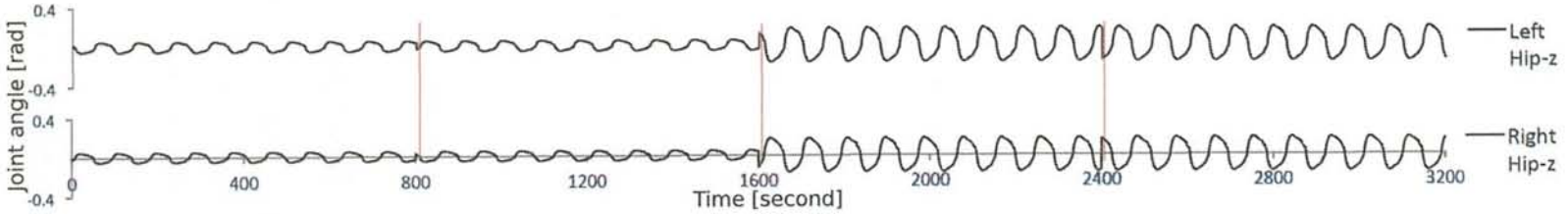



**Figure 5.11:** *The changing output of neural oscillator in hip-z joint while walking, using different direction parameters in certain time sampling*

#### 5.1.3.2 Application in Real Robot

We divided the real robot experiment into several processes. First, we ran the robot with different lengths of step by manipulating the gain parameter in order to implement various models of walking. When the value of gain increased, the length of step became longer. The experimental result from manipulating the value of gain is shown in Fig. 5.13

In the next experiment, we manipulated the direction parameter that represents the walking direction of the robot. By manipulating the direction parameter, we can adjust the direction of robot walking. When the direction was defined as "0", the robot walked in a straight direction. When the direction parameter was defined as greater than "0", the robot walked to the right. The robot walked to the left when the direction parameter was defined as less than "0". The results of this experiment are illustrated in Fig. 5.14. The variant walking direction and variant walking speed is a novelty in our approach, since other researchers did not consider it [142], [127], [118].

**5.1.3.2.1 Experiment in slope terrain** In order to emphasize the capability of the proposed locomotion system, we conducted an experiment in which the robot walked on a slope (Figs. 5.15a and 5.15b). In the first experiment, the surface has a $10^\circ$ tilt angle with horizontal direction. In this experiment, the locomotion generator parameters were adjusted according to the angle of the slope. Here, we changed the default angle of ankle-x joint ($\vartheta_{slope} = 0.1$). In the second experiment, the surface has a $14^\circ$ tilt angle with horizontal direction, with slope ankle joint 0.14 ($\vartheta_{slope} = 0.14$). Figure 5.16 shows the output for the joint angle in different slope terrains. There is a different joint angle in ankle-x in each slope terrain.

While the proposed locomotion in [142] enables walking on a slope surface with $11^\circ$ tilt angle with horizontal direction and the method in [118] enables walking on a slope surface with $10^\circ$ tilt angle with horizontal direction, our proposed locomotion can be implemented on a slope surface with $14^\circ$ tilt angle with horizontal direction.

### 5.1.4 Discussion

This research presented a new locomotion model by modifying the basic model of a neural oscillator. The locomotion model in this research consists of four types of neurons: 1) motoric neurons; 2) sensoric neurons; 3) command neurons; and 4) gain neurons. A multi-objective evolutionary algorithm was used effectively to construct the locomotion pattern by manipulating the weight of synapse between the motoric neurons. Based on the result from the evolutionary algorithm, the model acquired the optimum solution as the fastest walking speed of the robot, according to its capability, with low inertial oscillation.

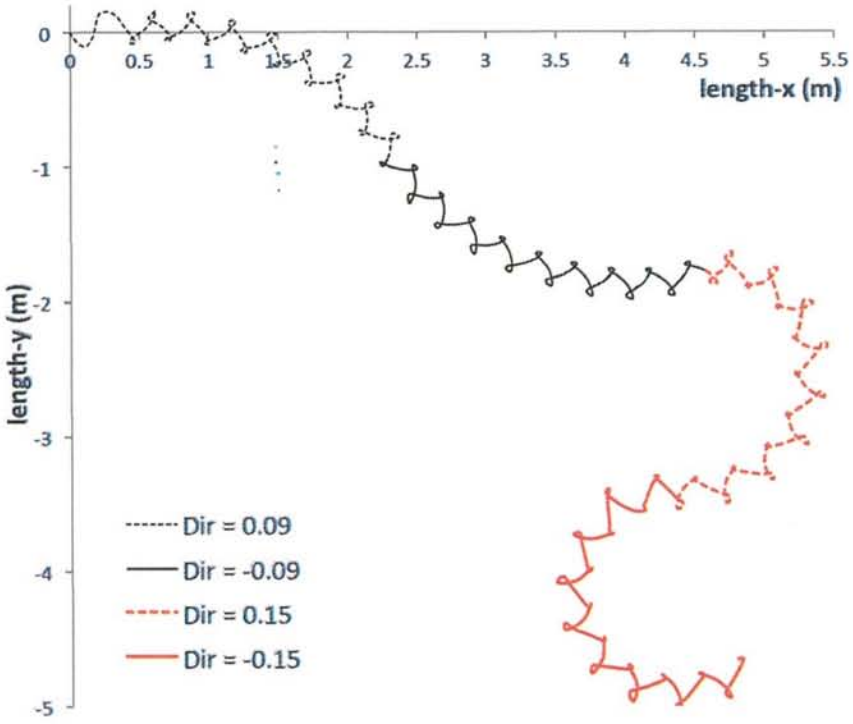



**Figure 5.12:** *Two-dimensional tracking from the center of mass in robot with different direction parameters*

In the proposed locomotion model, the control system for walking speed and walking direction was solved by determin- ing the parameter of the command neuron signal transmitted, as in our brain, to the gain neuron. The output from the coupled neuron oscillator is controlled by the gain neuron, depending on the signal from the command neuron. In the real robot experiment, various walking directions and walk- ing speeds were implemented as the result of the proposed system.

This research also presented a new stability model that sup- ports the locomotion system. The inertial sensor and ground touch sensor were installed to acquire the value of the sensoric neuron. We determined the synapse weight between the sensoric neuron and the motoric neuron dynamically by using RNN. In the experiment, the synapse weight between the sensoric neuron and the motoric neuron can be dynami- cally changed, depending on the environmental condition. The proposed model is proven to be effectively implemented for supporting locomotion, based on a Poincare phase diagram. In the diagram, the circular graphic was stable near the periodic orbit.

verall, this locomotion model was able to be implemented for dynamical locomotion. As for future work, we are going to design a passive dynamic control model for a stability system that supports a small sole of foot and has lower computational cost.

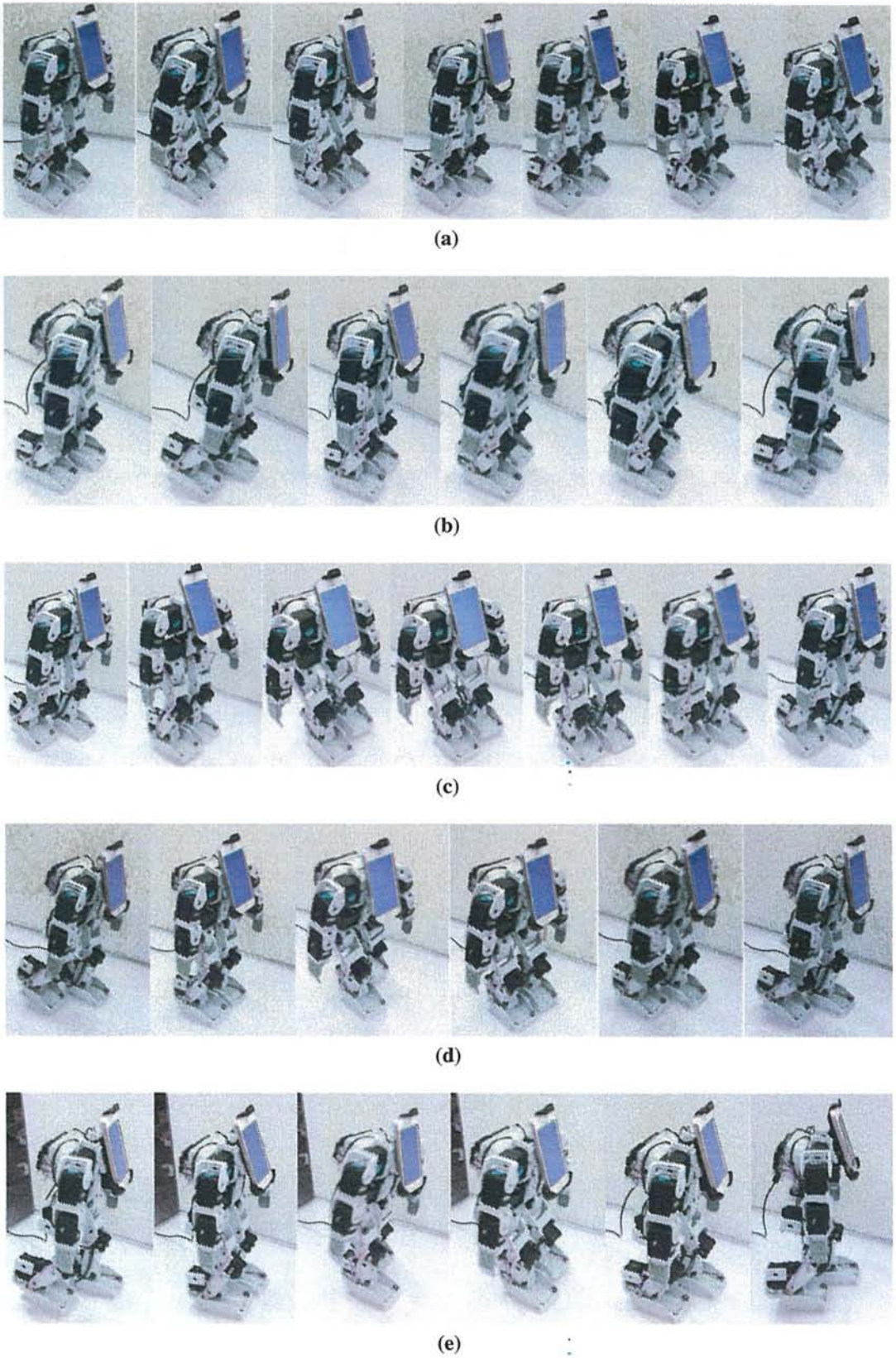



**Figure 5.13:** *Robot walking with different lengths of step. Direction parameter = 0 a) gain = 0.4; b) gain = 0.6; c) gain = 0.8; d) gain = 1.0; e) gain = 1.2*

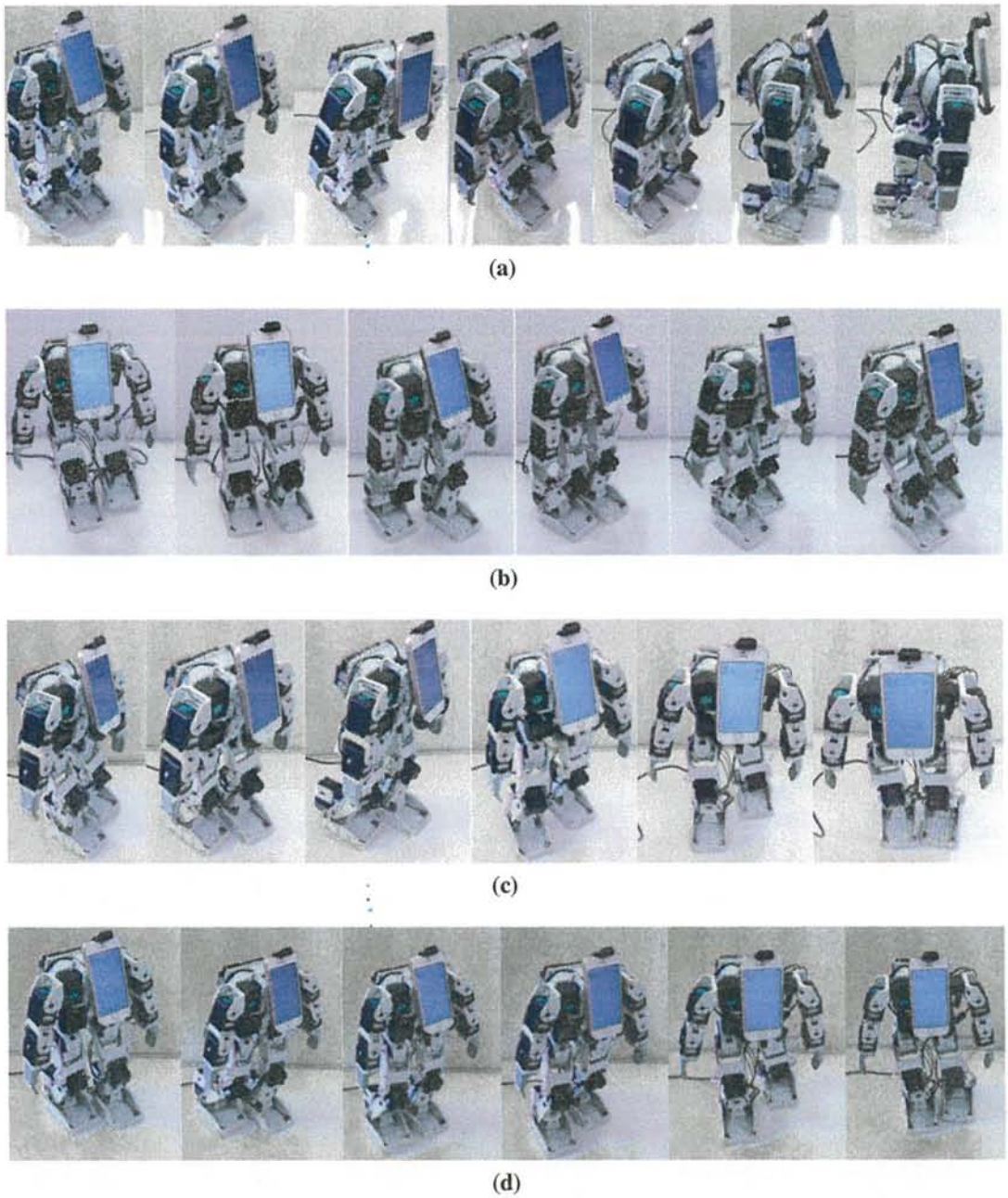



**Figure 5.14:** *Robot walking with different walking directions. Gain parameter = 0 a) direction = -0.15; b) direction = -0.09; c) direction = 0.15; d) direction = 0.09*

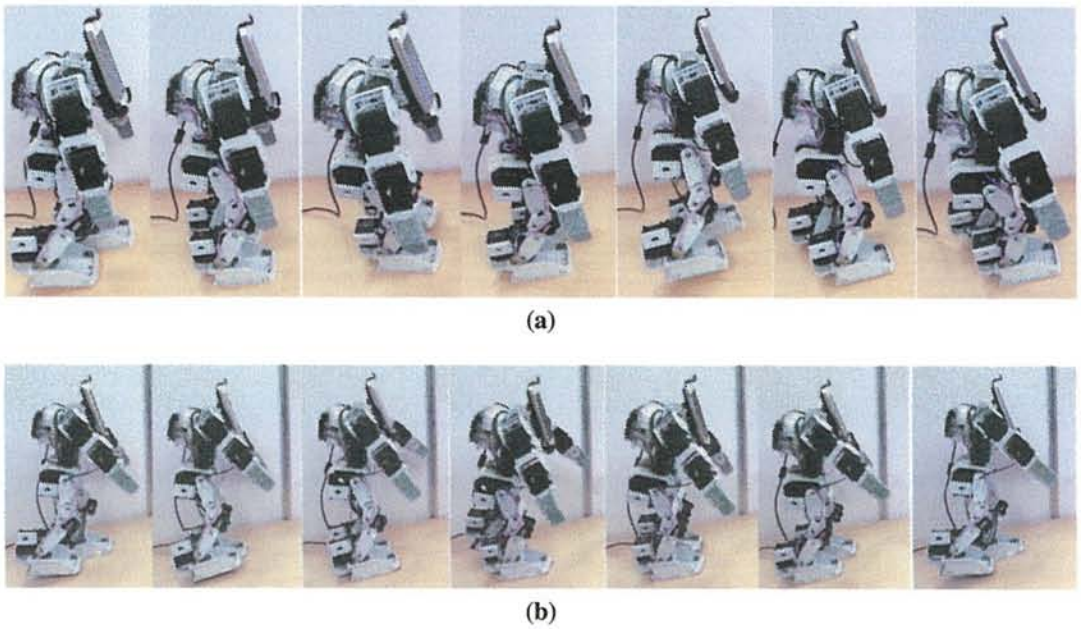


**Figure 5.15:** *Robot while walking on a slope: a) with* 10° *tilt angle; b) with* 14° *tilt angle*

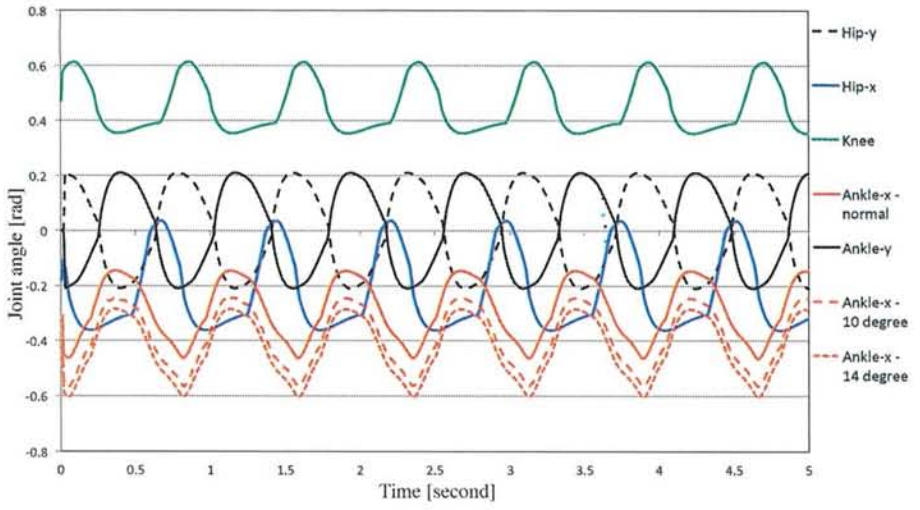



**Figure 5.16:** *The output of joint angle with different slope terrains*

## 5.2 Neural Oscillator based Locomotion (Quadruped robot)

Robotics technology, especially animal robot technology has increased rapidly in the world. In Japan, animal robots have a big demand especially from elderly people that need robot to accompany them in their daily activity. In the dynamic locomotion of animal robot, it is required to move in complex environments. In that way the robot can accompany elderly people in any environment. The interactions have mainly been in one direction, with robots taking inspiration from biology in terms of morphologies, modes of locomotion, and/or control mechanisms. In particular, many robot structures are directly inspired by animal morphologies, from snake robots, quadruped robots, to humanoid robots [86, 84, 85].

There are many methods to develop the locomotion in animal robot. Some researchers used physical approach and some researcher used biological approach to design the dynamic generation of rhythmic motion. Few researchers used central pattern generation (CPG) that has basis in neurophysiological studies. It is a type of neural network for the generation of rhythmic motion. Taga et al. designed stable and flexible locomotion realized as a global limit cycle generated by a global entrainment between the rhythmic activities of a nervous system composed of coupled neural oscillator [196]. In 2012, Baydin used central pattern generation to control bipedal walking mechanism. His research presents an approach where the connectivity and oscillatory parameter of a CPG network are determined by an evolutionary algorithm with fitness evaluations in a realistic simulation with accurate physics [22]. In 2014, Nassour et al. presented an extended mathematical model of central pattern generation in the spinal cord [142]. The proposed CPG model is used as the underlying low-level controller of humanoid robot to generate various walking patterns. Ren et al. also used central pattern generators model for dynamic locomotion. They developed malfunction compensation in six-legged walking locomotion [158]. The design and implementation of a general controller for quadruped locomotion allow the robot to use the whole range of quadrupedal gaits (i.e. from low speed walking to fast running). A general legged locomotion controller must integrate both posture and rhythmic motion control and must have the ability to shift continuously from one control method to the other according to the locomotion speed [133]. Maufroy et al. developed a general quadrupedal locomotion controller by using a neural model involving a CPG utilizing ground reaction force sensory feedback [133]. Ijspeert et al. used neural oscillator model to control the gait transition from swimming to walking and used different model of coupled neuron for walking and swimming locomotion and investigated different systems of coupled oscillators that can produce the typical swimming and walking gaits of the salamander [86, 84, 85, 83].

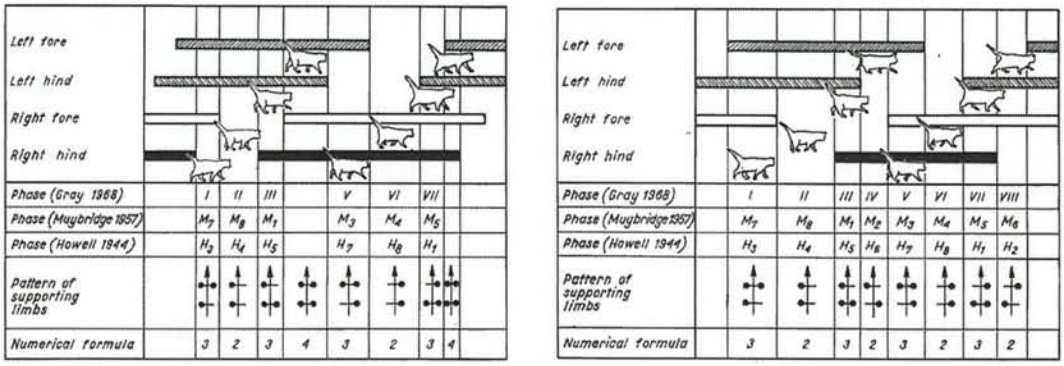



**Figure 5.17:** *Pattern of supporting limbs and numerical formula [10]*

In our previous research we dealt with conventional locomotion that used inverted pendulum and zero moment point approaches [176, 173]. We also implemented the Evolutionary algorithm for energy efficiency in that approach [177]. In this research, we develop the dynamic locomotion in four-legged animal robot based on neural oscillator. We investigate the coupled muscle in four-legged animal to get the mutual inhibitory and excitatory network in the neuron's structure. To optimize the synapse weight we use a multi-objective evolutionary algorithm. We designed the different neuron coupled structure in each locomotion model.

Our contribution to the locomotion model is, that we implement multi-objective evolutionary algorithm to optimize the best compromise between walking speed and stability. Other novelty in our research is, that we design inter-connection of neuron oscillator model in four-legged robot.

In locomotion of four-legged animals (cat and dog), longer swing and shorter stance duration for hind limbs, in comparison with the fore limbs, were observed. However, during trotting phases, the differences were insignificant. The correlations between swing and stance duration were found for both fore and hind limbs. As the walking animal moved faster, swing and stance duration became shorter. A walking cat moved at a rate of 1 to 4 km/hour, executing 12 to 6 strides. As the animal moved slower, more strides were performed. The numerical formula of the walking animal was 3-2-3-4-3-2-3-4 or 3-2-3-2-3-2-3-2 as depicted in Figure 5.17 [10].

### 5.2.1 Locomotion Generator

In this research we use 3 joints in each leg. We adjust the mechanical structure of dog illustrated in Fig. 5.18. The robot is equipped with ground reaction sensor and inertial sensor such as: Accelerometer, Gyroscope, Inertial Measurement Unit (IMU), and so on.

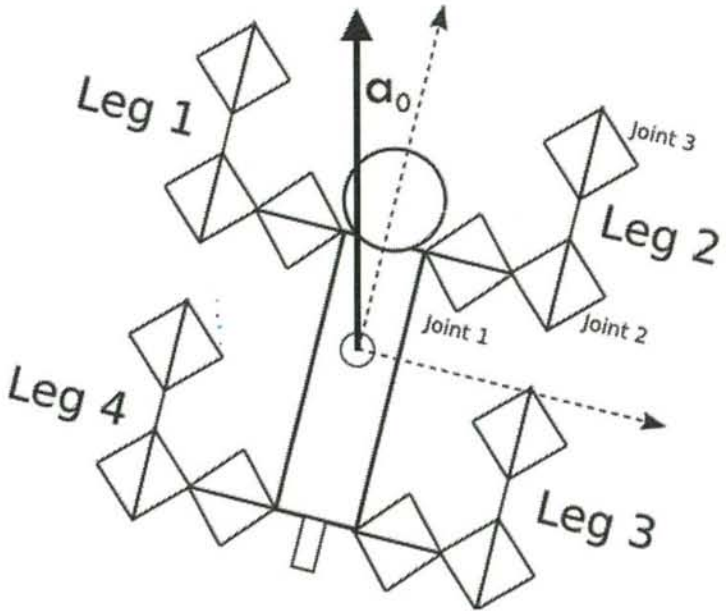



**Figure 5.18:** *Mechanical structure of the robot*

#### 5.2.1.1 Neural Oscillator

Before we design the locomotion generator system, we discuss the neural oscillator used as the element of the locomotion generator. We use neural oscillator generated by mutual inhibition between $n$ neurons with adaptation. The neural network model depicted in Fig. 5.30 generates oscillatory activity consisting of two tonically excited neurons, with the adaptation or self-inhibition effect, linked reciprocally via inhibitory connections. In neural oscillator, general mathematical model shown in Eqs. (5.23), (5.24), and (5.25) are presented by [130], [129].

$$\tau_l \dot{x}_i + x_i = -\sum_{j=1}^{n} w_{ij} y_j + S_i - b x'_i + x_0 \tag{5.23}$$

$$\tau_r \dot{x'}_i + x'_i = y_i \tag{5.24}$$

$$y_i = \max(x_i, 0) \tag{5.25}$$

where $x_i$ is the inner state of the $i$th neuron; $y_i$ is the output of the $i$th neuron; $x'_i$ is a variable representing the degree of adaptation or self-inhibition effect of the $i$th neuron; $b$ is a coefficient of self-inhibition effect; $x_0$ is an external input with a constant rate; $\tau_l$ and $\tau_r$ are time constant of the inner state and the adaptation effect, respectively.

In Eq. (5.23), $w_{ij}$ represents the strength of the inhibitory connection between the neurons, $\sum_{j=1}^{n} w_{ij} y_j$ represents the total input from the neuron inside a neural network, and $S_i$

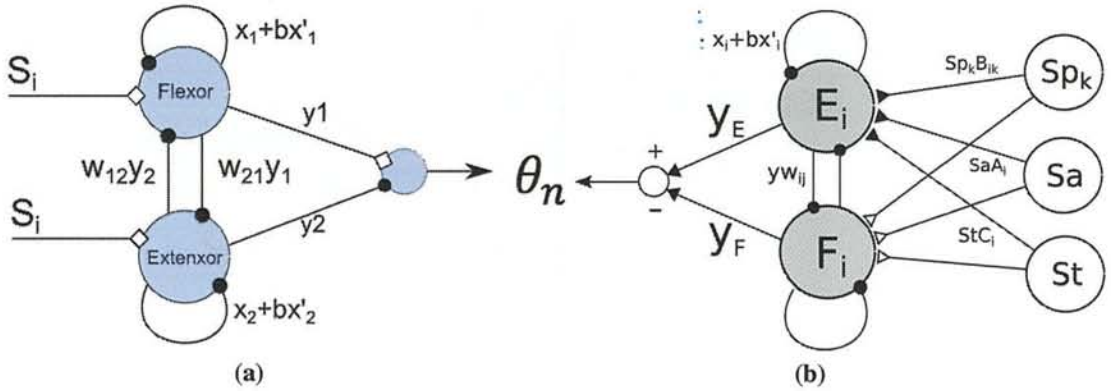



**Figure 5.19:** *(a) General coupled neuron (b) Coupled neuron with sensor connection*

is the constant value that represents the input from outside the neuron structure.

#### 5.2.1.2 Locomotion Generator

The motor neuron system receives two different types of sensory information: proprioceptive and exteroceptive information. In our model as depicted in Fig. 5.19b, we constructed the nonlinear oscillator equipped with 3 feedback sensors; $Sp_k$ is pressure sensor installed on each leg; $St$ is inclination sensor; $Sa$ is an acceleration sensor. In Eq. (5.26), $B_{ik}$, $A_i$, $C_i$, are constant values representing the influence of sensor to $i$th neuron and $m$ is the number of legs.

$$Sens(i) = \sum_{k=1}^{m} B_{ik} \cdot Sp_k + A_i \cdot Sa + C_i \cdot St \tag{5.26}$$

$$\tau_l \cdot \dot{x}_i = x_0 - x_i - \sum_{j=1}^{n} w_{ij} \cdot y_j + S_i - b \cdot x'_i - Sens(i) \tag{5.27}$$

$$\tau_r \cdot \dot{x}'_i + x'_i = y_i \tag{5.28}$$

$$y_i = \max(x_i, 0) \tag{5.29}$$

$$\theta_n = y_{2n} - y_{2n+1} \tag{5.30}$$

Equations (5.65), (5.66), and (5.67) were adapted from the neural oscillator model by Matsuoka [130], [129] explained in the previous subsection. Equation (5.68) explains 2 neurons (flexor and extensor) representing the union of the joint system. The robot has 12 degree of freedom, 4 legs, and each leg has 3 joints: knee joint, hip-x joint which is rotational about the x axis, and hip-y joint which is rotational about the y axis. We make some limitation

in each joint: $\pi/2 > \theta_1^i > -\pi/2$, $0 > \theta_2^i > -\pi/2$, $\pi/4 > \theta_3^i > -\pi/4$. In our neuron structure we have 24 neurons illustrated in Fig. 5.20. We investigate the muscular structure and relationship between neurons. The robot has its main circular neurons in $E_1$, $E_3$, $E_5$, $E_7$ as the server neurons, while other neurons are the client neurons. Server neurons generate the oscillator signal to the client neurons.

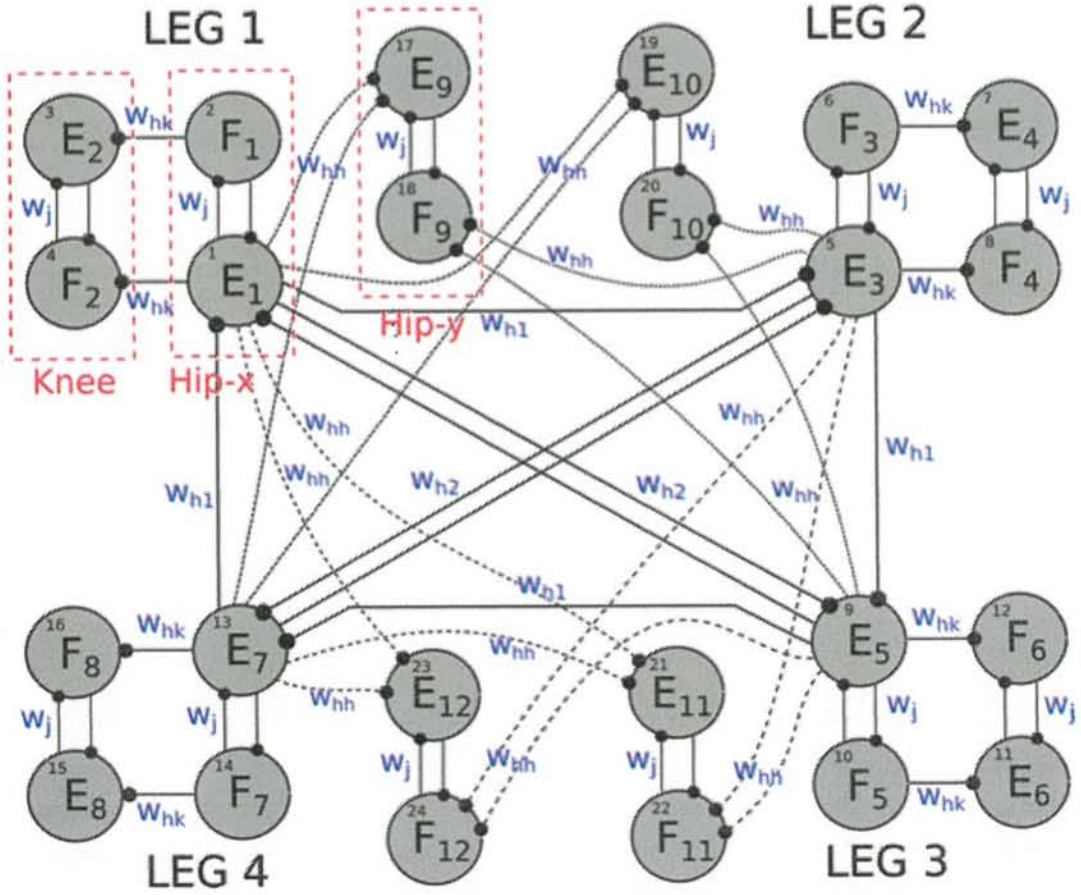



**Figure 5.20:** *Diagram of Neuron Structure*

In the synapse weight between motor neurons shown in Fig. 5.20, we separated weight synapses into 5 part, in one part having the same weight value: $w_j$ is the weight of synapse between flexor and extensor neuron in the same joint; $w_{hk}$ is the weight of synapse between hip-x neuron and knee neuron; $w_{h1}$ is the circular weight in the main neuron; $w_{h2}$ is the diagonal weight in the main neuron; and $w_{hh}$ is the weight between hip-x neuron and hip-y neuron.

The feedback pathway is designed by adapting the human-like mechanism. The connection between motor neuron and sensory neuron can be seen in Fig. 5.21, where $Sp_k$ is the pressure sensor in the $k$th leg and has $B_{ik}$ as the weight parameter between the pressure sen-

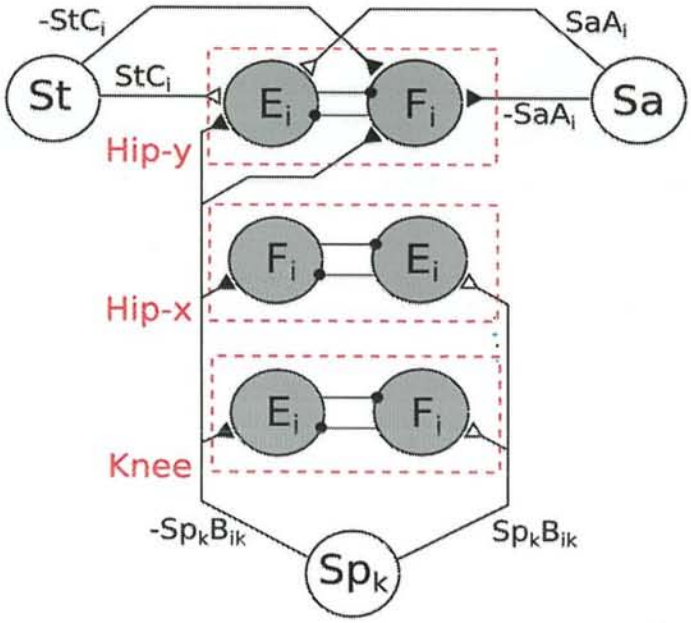



**Figure 5.21:** *Sensor connection*

sor in the $k$th leg and the $i$th motor neuron. Pressure sensor sends negative signal to extensor neuron in knee joint, flexor neuron in hip-x joint, and neuron in hip-y joint. It also sends positive signal to flexor neuron in knee joint and extensor neuron in joint hip-x. $Sa$ is an acceleration sensor that detects the acceleration oscillation of the robot movement. $A_i$ is the weight parameter between sensor $Sa$ and the $i$th neuron. $St$ is a tilt sensor that detects the angle of the robot body and has $C_i$ as its weight to the $i$th neuron. Not all motor neurons are connected to certain sensors. Both $Sa$ and $St$ send negative signal to flexor neuron in hip-y joint and send positive signal to extensor neuron in hip-y joint. The structure connection and the feedback signal from movement speed sensor, tilt, acceleration sensor are expected to be effective feedback for stability and adaptive locomotion.

### 5.2.2 Synapse Optimization

There are several methods for solving multi-objective optimization problems. The methods can be divided into two main categories: Pareto front based methods and weighting factors based methods. One of the main drawbacks of the weighting factors based methods is to choose appropriate weighting values for transforming the multi-objective problem into a single objective problem. On the other hand, in the case of Pareto front based method, the problem is not transformed into a single objective problem. The goal in this case is to obtain the Pareto front which contains equally optimal solutions for the given problem from the

multi-objective point of view.

The development of locomotion based on neural oscillator implies the optimization of the oscillation of body tilt $\bar{\sigma}$ (minimization), the rate of change of direction (minimization), and the velocity of movement $\bar{v}$ (minimization). We calculate the body tilt, change of direction, and velocity by Eqs. (5.33), (5.34), (5.35) by using computer simulation. The fitness function of this system is computed by Eqs. (5.56), (5.57). In Neural Oscillator structure, the weight of synapse between neurons should be determined so as to have the desired velocity with minimum oscillation of body tilt. To get the fitness value we get the locomotion simulation run by using ODE and analyze the output sensor. We normalize the data for fitness value calculation using absolute value for $\sigma$ and non-absolute value for $\mathrm{L}, \mathrm{x}, \mathrm{y}$.

$$\sigma(t, w_{ij}) : |\mathrm{R}| \tag{5.31}$$

$$\mathrm{L}(t, w_{ij}), \mathrm{x}(t), \mathrm{y}(t) : \mathrm{R} \tag{5.32}$$

In Equations (5.31) and (5.32), $\sigma$ is the body tilt in absolute value and L, $\mathrm{x}(t)$, $\mathrm{y}(t)$ are the length of movement, length of movement in x axis, and length of movement in y axis. Those are resulted in a time sampling.

$$\bar{\sigma}_{(w_{ij})} = \frac{1}{T}\sum_{t=0}^{T}\sigma(t, w_{ij}) \tag{5.33}$$

$$\bar{v}_{(w_{ij})} = \frac{1}{T}\sum_{t=0}^{T}\frac{\mathrm{d}}{\mathrm{dt}}L\big((t, w_{ij})\big) \tag{5.34}$$

$$\bar{a}_{(w_{ij})} = \frac{1}{T}\sum_{t=0}^{T}\frac{\mathrm{d}}{\mathrm{dt}}\Big(\sqrt{\dot{x}^2 + \dot{y}^2}\sin\Big(\alpha_0 - \arctan\Big(\frac{\dot{x}}{\dot{y}}\Big)\Big)\Big) \tag{5.35}$$

$$w_{ij}, A_i, B_{ik}, C_i a = \underset{w_{ij}, A_i, B_{ik}, C_i}{\arg\min}\ a(\bar{\sigma} + \bar{a}) \tag{5.36}$$

$$w_{ij}, A_i, B_{ik}, C_i a = \underset{w_{ij}, A_i, B_{ik}, C_i}{\arg\max}\ a\bar{v} \tag{5.37}$$

The chromosome structure in this algorithm is composed of 5 weight values between motor neurons ($w_{ij}$); 4 weight values of $B_{ik}$; 2 weight values of $A_i$; and 2 weight values of $C_i$. Each parameter is represented by one gene, thus one individual has 13 genes, since there are 13 parameters to be optimized.

### 5.2.3 Experimental Result

In this research, we simulated the system by using Open Dynamics Engine computer simulation. We designed the robot with parameters as: weight, height, center of mass, environment parameter etc.

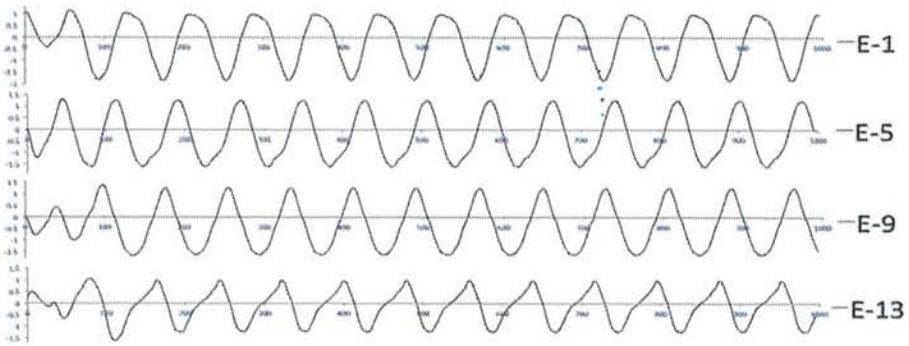



**Figure 5.22:** *Signal Angle of Joint*

To develop the locomotion pattern that is suitable to cat locomotion, at first, we tried to do hand-tuning approach to get the weight of synapse parameter in the main locomotor (4 neurons) that produced the pattern as shown in Fig. 5.22. After that we designed the coupling neuron between main neuron and client neuron and designed the coupling system between motor neuron and sensory neuron as shown in Fig. 5.20 and 5.21, respectively.

**Table 5.6:** *NSGA-II Parameters*

| Parameter | Exp. 1, Exp. 2, Exp. 3 |
|---|---|
| Population size | 12 indv., 32 indv., and 64 indv. |
| Num. of Generations | 500 Gen., 200 Gen., 100 Gen. |
| Num. of Objectives | 2 obj.: Speed and Stabilization |
| Chromosome | 13 real numbers |
| Crossover and mutation prob. | 0.3 and 0.3 |

The parameter setting of NSGA-II is summarized in Table 5.15. We analyze the change of walking speed and stabilization. In Fig. 5.24a we tried the first experiment and compared the Pareto front of 50-th, 100-th, and 500-th generations. Until the 500-th generation, we got 3 best individuals those have 0.104, 0.132, and 0.172 values in oscillation of stabilization and 1.983, 1.956, and 1.91 values in the velocity of movement.

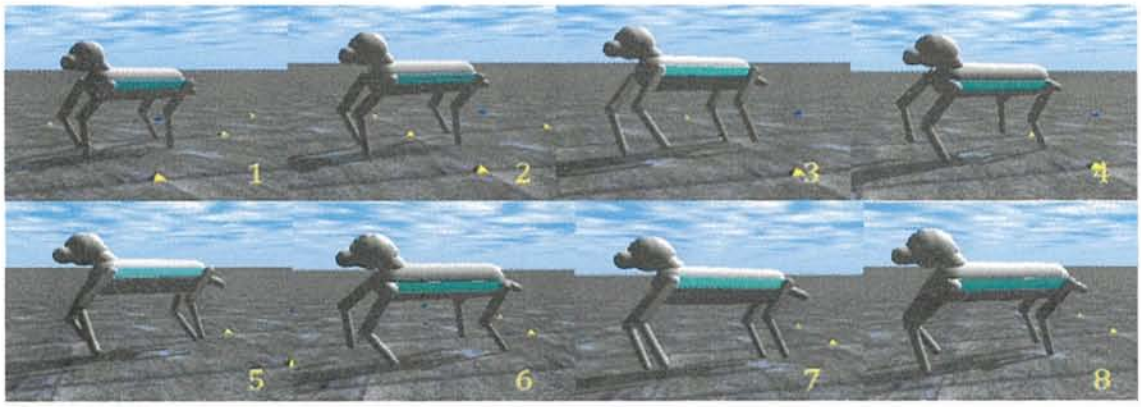



**Figure 5.23:** *Simulation Result*

We tried to increase the number of individuals as shown in Fig. 5.24b and 5.24c. We tried 32 individuals until 200 generations and 64 individuals until 100 generations. The spreading of individuals with 64 individuals is better than with 32 individuals. With 64 individuals, we get the maximum Pareto front in the 100-th generation.

**Table 5.7:** *Result of Weight Parameters*

| Parameter | Value |
|---|---|
| $w_{hk}$ | 0.738 |
| $w_j$ | 2.084 |
| $w_{hh}$ | 0.213 |
| $w_{h1}$ | 1.7221 |
| $w_{h2}$ | 1.1100 |
| $B_{1,1}$; $B_{5,2}$; $B_{9,3}$; $B_{13,4}$ | 0.891 |
| $B_{2,1}$; $B_{6,2}$; $B_{10,3}$; $B_{14,4}$ | 0.789 |
| $B_{3,1}$; $B_{7,2}$; $B_{11,3}$; $B_{15,4}$ | 2.248 |
| $B_{4,1}$; $B_{8,2}$; $B_{12,3}$; $B_{16,4}$ | 1.031 |
| $A_{17}$; $A_{19}$; $A_{21}$; $A_{23}$ | 0.674 |
| $A_{18}$; $A_{20}$; $A_{22}$; $A_{24}$ | -0.343 |
| $C_{17}$; $C_{19}$; $C_{21}$; $C_{23}$ | -0.919 |
| $C_{18}$; $C_{20}$; $C_{22}$; $C_{24}$ | 2.396 |

We chose the parameters resulted from the individual which has faster speed (0.72; 1.91) that are tabulated in Table 5.7, where the value of weights between each neuron is shown. The other value of weights not included in Table 5.7 are zero. The simulation running process is depicted in Fig. 5.45. In Fig. 5.25, we show the evolution of the fitness objective 1 and fitness objective 2. Before the 25-th generation, the fitness has high gradient and after the

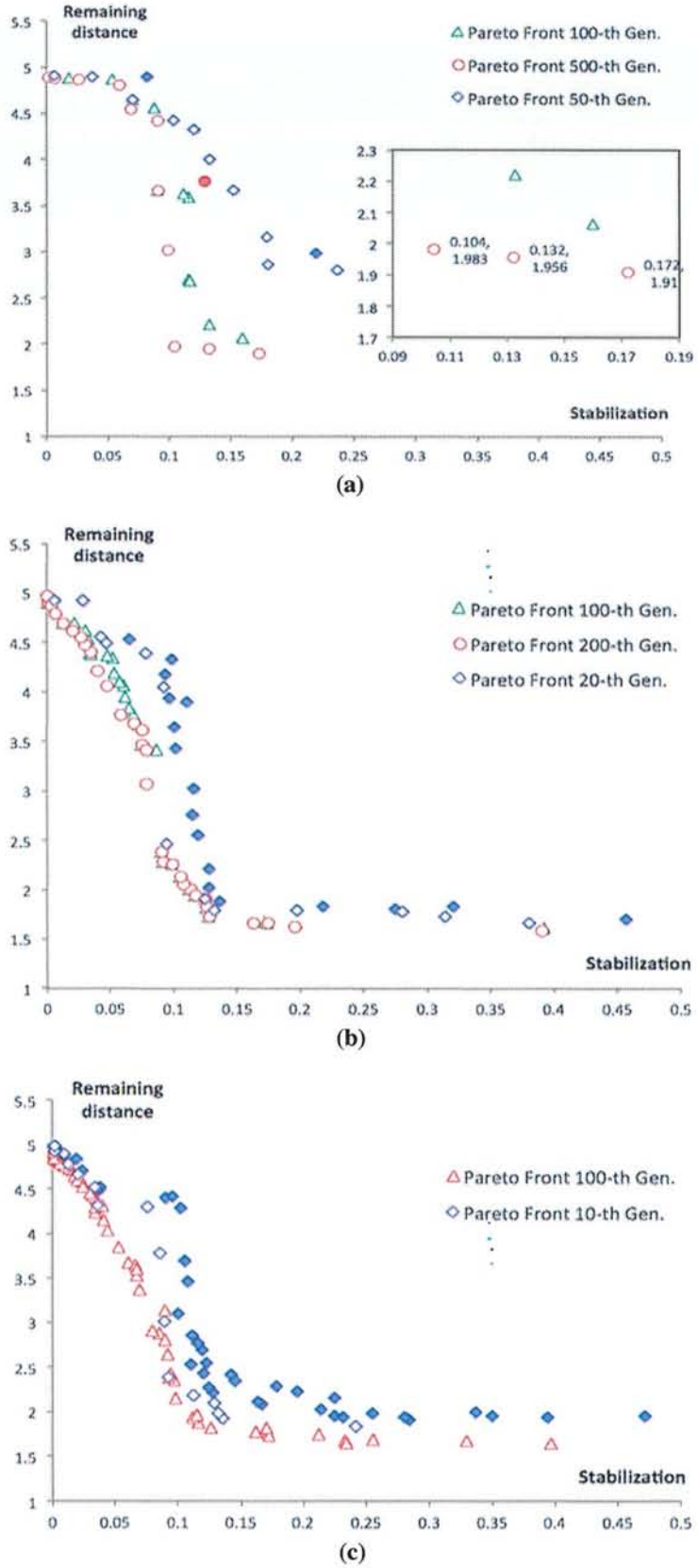



**Figure 5.24:** *Pareto front a) 12 individuals b) 32 individuals c) 64 individuals*

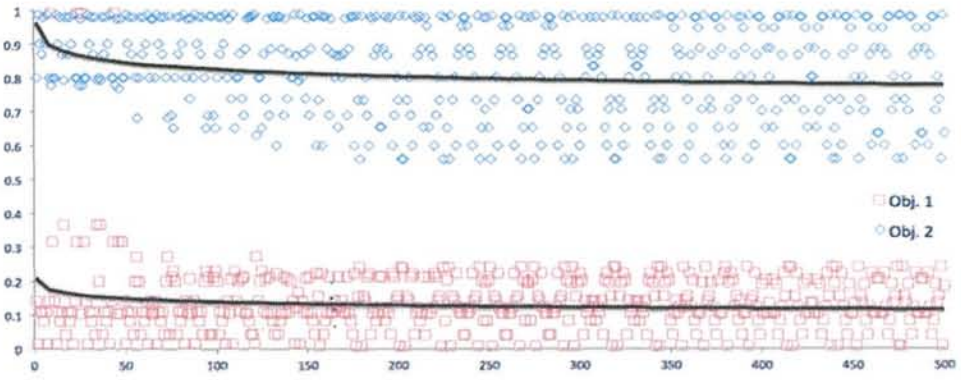



**Figure 5.25:** *Fitness evolution*

25-th generation, it becomes steady state. The robot has the maximum performance in the 500-th generation.

The result of angles in each joint is shown in Fig. 5.26. One signal resulted from 2 coupled neuron, extensor and flexor. The signal can dynamically adjust to the environmental conditions. The signal is influenced by ground reaction sensor and body inclination feedback. While the foot touches the ground, the sensory neuron sends the signal to the joint neuron. Negative signal will be sent to flexor neuron and positive signal will be sent to extensor signal.

By using this system, locomotion can be dynamically adjusted to the environment. However, malfunctioned leg cannot be supported by this system. For this robot, we are going to develop the system that can support malfunctioned leg condition. We try to optimize the synaptic weight by using learning system based on recurrent neural network backpropagation through time.

### 5.2.4 Conclusion

In this research we designed the locomotion of a four-legged robot based on neural oscillator. We designed the coupling mechanism between motor neurons and between motor and sensory neuron. We used multi-objective evolutionary algorithm to optimize the weights of the synapses. The weights of the synapses can be optimized well with two objectives: velocity and stabilization (tilt and acceleration). Non-domination fitness is resulted so that we get good speed of the movement with oscillation that can be accepted by robot mechanism. The locomotion can be adaptive with ground reaction and oscillation movement.

In the future research after we complete the locomotion in four-legged animal robot, we will design the adaptive locomotion of humanoid biped robot based on neural oscillator that is combined with recurrent neural network for hand reaction system to support the stabiliza-


**Figure 5.26:** *Signal Angle of Joint*

tion in movement. Recurrent neural network will be implemented for learning the synapse weight between sensory neuron and motor neuron similarly as in human walking.

## 5.3 Bezier Curve Model for Efficient Locomotion Model

This section presents Bezier curve based passive neural control applied in bio-inspired locomotion in order to decrease the computational cost implemented for 4 legged animal robot which has 3 joints in each leg. Neural oscillator model is applied for generating the walking pattern in bio-inspired locomotion. Bezier curve based optimization represents passive neural control supported by evolutionary algorithm for representing the relationship equation between neuron signal and reference joint signal. Passive neural control is implemented in order to reduce the neuron complexity in neuro-based locomotion by controlling 3 joints with one signal without decreasing the performance both in walking pattern and in its stability level, whereas one leg is represented by one motor neuron. Therefore, the 4 legged robot is controlled by 4 motor neurons which have feedback connection with ground and inertial sensor. In order to prove the effectiveness, we implemented the model in computer simulation and in a small 4 legged robot. This model can decrease the computational cost so it is possible to apply the model in either animal or humanoid robot with low frequency processor.

### 5.3.1 Introduction

After a disaster, the environments often have terrains that are unstructured, diverse and challenging. At the same time, human rescue team deployment is impeded by the potential danger. Robot is a good substitute for such rescue deployment. The goal of these robots is to provide first responders with the ability to remotely access and survey the disaster zone, locate victims, and possibly manipulate the environment. Wheeled robots are the simplest choice, but they have limitations in disaster environment. Alternative to that are walking robots. The walking robot as a movable working platform should remain stable while standing. Therefore, such robots have to have more than three legs to maintain stability. Stability margin increases as the number of legs increases, up to the point when there are 8 legs [76].

The computer the robot carries should be light-weight, and thus, might be limited in terms of its processing power and frequency. Such processor is used to compute the kinematics of quadruped robot locomation, where computational power requirement is aggravated by the need to deal with rough terrain and surveillance of the surroundings. It is therefore desirable to increase computational efficiency by decreasing the processing power of kinematics pro-

cessing. In the rescue robot development, cost is an important factor, where cutting it comes at a price of lower computational cost [23].

Bio-inspired locomotion has proven to perform well as dynamic walking generator. Several bio-inspired locomotion models have been proposed by several researchers [86, 84, 196, 136, 146]. The locomotion based on neural oscillator has been implemented in four legged robot in order to acquire the dynamical locomotion [86], [84], [85]. Beside that, Ijspeert et al implemented neural oscillator for passive compliant knee joint in Cheetah-like robot [165]. Ijspeert et al also developed neural oscillator based gait transition in quadruped robot by forming the interlimb connection [86]. But, Owaki et al only considered interlimb coordination without neural connections in their gait transition model for quadruped robot [146]. In previous development, we also developed bio-inspired locomotion in four-legged robot [178]. The dynamical locomotion implies the complexity of the neural structure. The complexity of neural network is able to generate various dynamic walking patterns. The high computational cost will be caused in order to implement the high complexity neural structure and bio-inspired dynamic locomotion. Based on the previous analysis, the high specification device has high frequency based processor required for implementing the bio-inspired locomotion.

In biomechanics study, the configuration on muscle tendon in each joint enables to generate the finger joints by one signal although there are at least 3 joints in the human finger [199]. The muscle tendon is generated by neuro signal in order to move the joint in the finger [17]. Based on this bio mechanism model, proposed system generates 3 joints of each leg by one signal neuron. Therefore, only four neurons are required for generating four legged robot locomotion. Different from underactuated joint model, this proposed system is neural signal based locomotion that can dynamically change the joints trajectory.

The contribution of our proposed model is to reduce the complexity of the neural based locomotion, thus reducing computational complexity, without decreasing the locomotion performance by implemented passive neural model. Other novelty in this proposed system is using Bezier curve model [73] optimization in order to optimize the relationship graph between neural signal and reference joints (hip-y, hip-x, and knee).

### 5.3.2 Neuro-based Locomotion System

Our bio-inspired locomotion uses neural oscillator model proposed by Matsuoka [130]. This model is generated by mutual inhibition between certain neurons. Each neuron also acquires adaptation signal input. In our previous research, the number of required neurons was too big, there were 24 motor neurons to represent 12 joints [178]. That neuron complex-

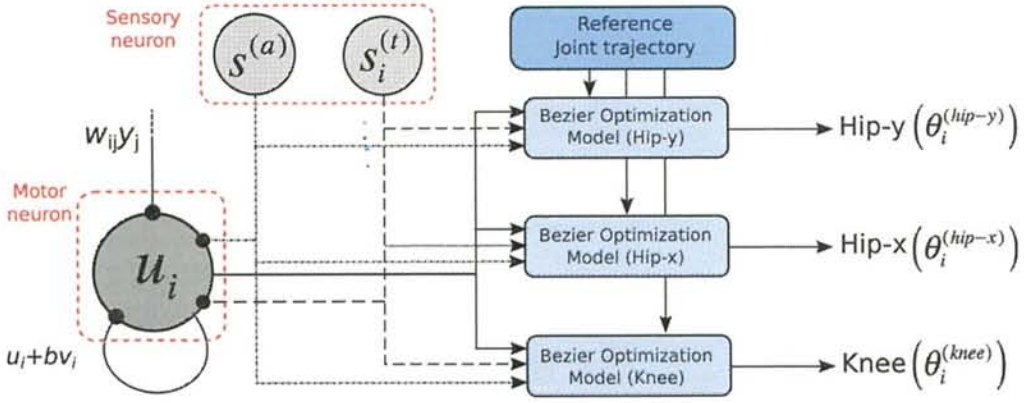



**Figure 5.27:** *Locomotion model in each leg composed of one neuron. One motor neuron generates 3 joints based on the Bezier optimization model. Sensory neurons are providing the feedback signal for dynamic locomotion.*

ity affected the computational cost in the implementation. This proposed model reduces the number of neurons to 4 for generating the angle value of 12 joints shown in Fig. 5.28. The design of neuron interconnection can be seen in Fig. 7.2. Four neurons are used for implementing sensor integration, therefore the neuron can generate the signal phase independently depending on the sensor feedback. The mathematical model of the proposed neuro-locomotion system is shown in Eqs. (5.65), (5.66), (5.67).

$$\tau \dot{u}_i = (u_0 - u_i - \sum_{j=1}^{n} w_{ij} y_j - b v_i)\tau_f \tag{5.38}$$

$$\tau' \dot{v}_i = (y_i - v_i)\tau_f \tag{5.39}$$

$$y_i = \max(u_i, 0) \tag{5.40}$$

$$u_i' = u_i g_i \tag{5.41}$$

where $u_i$, $y_i$, and $v_i$ are the inner state, output value, and a variable that represents the adaptation value or the self-inhibition effect of the $i$th neuron, respectively; $b$ is the rate of the adaptation value. The external input for coupled neurons, which has a constant rate, is denoted by $u_0$. The time constant of the inner state and the adaptation effect in the neuron are represented by $\tau$ and $\tau'$, respectively. In Eq. (5.65), $w_{ij}$ represents the strength of the inhibitory effect between the motor neurons that is optimized offline; $\sum_{j=1}^{n} w_{ij} y_j$ represents the total of the signal input from the neuron. In Eqs. (5.65) and (5.66), $\tau_f$ is used for controlling the frequency of oscillation. In Eq. (5.68), $g_i$ is the dynamic parameter representing the gain of the neuron signal in the $i$th neuron. This parameter carries the responsibility for adjusting the amplitude of the neuron signal.

In Eq. (5.42), the joint angle value is represented by relationship equation formed by

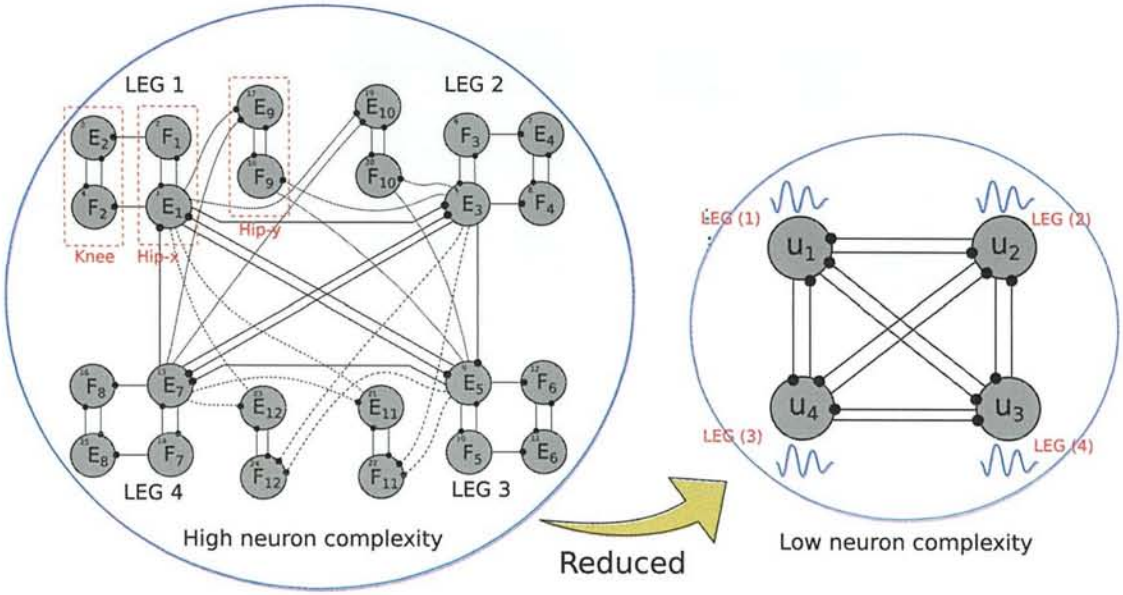



**Figure 5.28:** *The previous structure of neural oscillator that has 24 motor neurons connected to each other is decreased into 4 motor neurons*

Bezier curve [73] based optimization explained in the next subsection, where $p$ represents either hip-y, hip-x, or knee joint. The joint angle values depend on the input from the $i$th neuron signal in each leg and the influences from the sensory neurons ($D_i^{(p)}$) are calculated in Eq. (5.43).

$$\theta_i^{(p)} = B_i^{(k)}(t(u_i')) - D_i^{(p)} \tag{5.42}$$

$$D_i^{(p)} = \psi^{(p)} s_i^{(t)} + \chi^{(p)} s^{(roll)} + \gamma^{(p)} s^{(pitch)} \tag{5.43}$$

where $\psi^{(p)}$, $\chi_{i,p}$, $\gamma_{i,p}$ are the parameters representing the gain of ground touch ($s_i^{(t)}$), tilt body angle sensory neuron in roll direction ($s^{(roll)}$), and pitch direction ($s^{(pitch)}$), respectively. The gain parameters are set based on preliminary analysis where $\psi^{(p)} \in \{0.00; 0.52; 0.45\}$, $\chi^{(p)} \in \{0.56; 0.00; 0.00\}$, and $\gamma^{(p)} \in \{0.00; 0.5; 0.35\}$.

### 5.3.3 Bezier Curve Optimization for Passive Neural Control

The angle value of the joints are generated from the effect of output neuron signal. Therefore, the value of hip-x, hip-y, and knee joints are affected by the input from neuron signal in each leg. The value of angle joint in the $i$th leg ($\theta_p^i$, where $p \in$ {hip-x, hip-y, knee}) depends on the neuron value of neuron $u_i'$. Active neuron signal is generated by mutual inhibition of coupled neuron while angle joint is generated by an equation that describes the relationship between signal neuron component and certain joint angle. In order to acquire the relationship model, we record the required data from the previous model of locomotion, where all joints

are represented by coupled neuron [178]. Its computational cost will be reduced by using this proposed model. The relationship graph is tabulated in Fig. 5.33.

Bezier model in locomotion is usually used as the kinematics [73]. It is also proposed as the method to solve the kinematic problem of hand-eye coordination, and more specifically, tool-eye recalibration of humanoid robots [205]. Bezier curve model generates smooth signal which is easy to be adjusted into the graph reference [73]. In this proposed model, we used a 2-dimensional quadratic Bezier curve model as the optimization function in order to get the relationship equation between the angle value of joints and motor neuron. The mathematical model of Bezier curve model is presented in Eq. (5.44), where the Bezier curve point in time $t$ is denoted by $\mathbf{B}(t)$; $t$ is from 0 to 1; $n$ represents the number of degrees in Bezier curve and $\mathbf{P}_i$ is the $i$th selection point.

In Bezier curve, the axis value ($x$ axis and $y$ axis) is resulted from 2 Bezier point notations which are $B_x(t)$ representing neural signal and $B_y(t)$ representing joint angle with $t$ as the timing control. In this case, we have to find $y$ axis $B_y$ (passive neuron) by determining the value of $x$ axis $B_x$. Parameter $t$ is required for relating both parameters. Therefore, in order to acquire the value of $t$, we invert the equation of $B_x(t)$ to become $t(B_x)$. Since $B_x(t)$ parameter is a third order polynomial equation, we used Cardano formula to solve the inverse equation. After inversion, the Bezier point in $y$ axis can be denoted as $B_y(t(B_x))$.

$$\mathbf{B}(t) = \sum_{i=0}^{n} \binom{n}{i} t^i (1-t)^{n-1} \mathbf{P}_i \tag{5.44}$$

One relationship graph represented as 3 quadratic Bezier curves is illustrated in Fig. 5.29. We use time parameter in order to facilitate and support the model and separate it into 3 curves. In the graph generated from $t = 0$ to $t = t_{max}$, when $0 \leq t < t_1$ curve 1 is generated, when $t_1 \leq t < t_2$ curve 2 is generated, when $t_2 \leq t < t_{max}$ curve 3 is generated.

Based on the model, we have 12 two-dimensional points to be optimized. From the preliminary studies, we used Bacterial Memetic Algorithm (BMA) to optimize the Bezier point. More details about the algorithm can be found in [25]. The bacterium structure has 8 genes divided into 2 parts, genes representing the point in the line of reference ($G_1^{(b)}$, $G_2^{(b)}$) and genes representing the point supporting the curve ($G_3^{(b)}$ - $G_8^{(b)}$). $G_1^{(b)}$ represents points $P_3$ and $P_4$, and $G_2^{(b)}$ represents points $P_7$ and $P_8$. Other genes $G_3^{(b)}$, $G_4^{(b)}$, $G_5^{(b)}$, $G_6^{(b)}$, $G_7^{(b)}$, $G_8^{(b)}$ represent points $P_1$, $P_2$, $P_5$, $P_6$, $P_9$, $P_{10}$, respectively. In the encoding process which is presented in Eq. (5.45), parameter $G_1^{(b)}$ and $G_2^{(b)}$ are decided depending on the point in reference graph with certain time cycle $t$. Therefore, the limitation point is from 1 to the number of time cycles $t_{max}$. Other genes are decided by the position of the point in reference line. $G_3^{(b)}$ and $G_8^{(b)}$ depend on the point position of $P_0$, $G_4^{(b)}$ and $G_5^{(b)}$ depend on the point position of $P_3$,

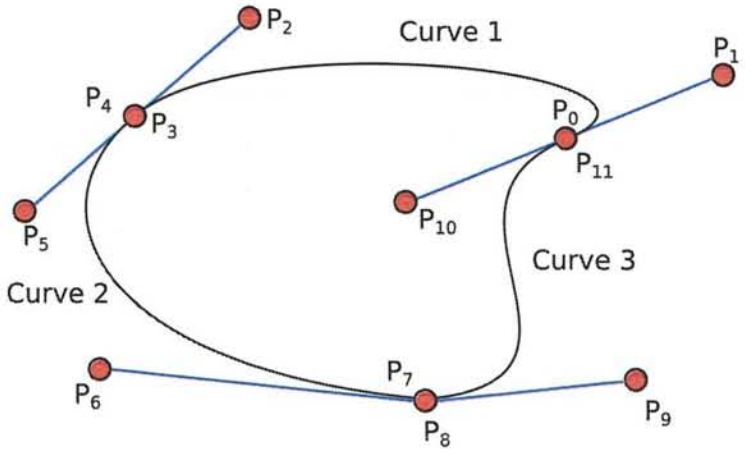



**Figure 5.29:** *The proposed Bezier curve model. There are 3 quadratic Bezier curves, where 4 points are required in each curve, $P_0 - P_3$, $P_4 - P_7$, $P_8 - P_{11}$ represent 1st curve, 2nd curve, 3rd curve, respectively. $P_3$, $P_7$, $P_{11}$ are equal to $P_4$, $P_8$, $P_0$, respectively. $P_0$ and $P_{11}$ are defined as the first point of the joint trajectory. Therefore, there are 8 genes to be optimized which consist of 2-D parameters.*

and $G_6^{(b)}$ and $G_7^{(b)}$ depend on the point position of $P_7$.

$$G_k^{(b)} = \begin{cases} R(t) + \alpha r(g_{max} - g_{min}) - g_{min} & \text{if } 1 \leq k \leq 2 \\ t = 1 + r(t_{max} - 1) & \\ P_l^{(x,y)} + \beta r(g_{max} - g_{min}) - g_{min} & \text{if } 3 \leq k \leq 8 \end{cases} \tag{5.45}$$

In Eq. (5.45), $P_l^{(x,y)}$ is a 2-dimensional point of Bezier model, where, if $k$ equals 3 and 8 then $l$ equals 0, if $k$ equals 4 and 5 then $l$ equals 3, and if $k$ equals 6 and 7, then $l$ equals 7. $R(t)$ is a 2-dimensional coordinate of reference graphic with input $t$. In Eq. (5.45), $r$ is a random value from 0 to 1; $g_{min}$ and $g_{max}$ are the minimum and maximum value from $\{G_3^{(b)}, \ldots, G_8^{(b)}\}$ set; $\alpha$ and $\beta$ are constant values where $\alpha \ll \beta$.

In the evaluation process, we minimize the error of the graph in each curve ($E_c$), where the $c$ parameter represents the index of the $c$th curve as depicted in Eq. (5.46). The error ratio ($E_1 : E_2 : E_3$) determines the probability of BMA operation, where e.g., if the percentage error of the 1st curve is high, then the genes representing the 1st curve has high probability for modification. The total fitness is accumulated from $E_1$, $E_2$, and $E_3$

$$E_c = \sum_{j=t_{c-1}}^{t_c} \Big( |R_x(j) - B_x(j)| + |R_y(j) - B_y(j)| \Big). \tag{5.46}$$

### 5.3.4 Experimental Result and Discussion

In the experimental result, we analyze the walking pattern of cat [10] and record the joint angles which are generated based on orbital function based swing trajectory proposed in [169]. We represent the cat having 3 joints (hip-x, hip-y, and knee) in each leg as depicted in Fig. 7.7, where the 4 sequences for 1 walking cycle is as follows: first swing is Leg 2, second is Leg 4, third is Leg 3, and the forth swing is Leg 1. The reference joint angles for each leg can be seen in Fig. 5.34.

#### 5.3.4.1 Neural Oscillator Optimization

Here, we reduce the number of neurons required for the walking pattern from 24 neurons to 4 neurons. In order to form the neural signal appropriate with the reference of the joint trajectory and to have the same phase difference, the synaptic weight values of 4 neurons representing the active signal are optimized by using BMA that has proven its capability in the preliminary tests.

The aim of this optimization is to form the suitable signal for 4 legged robot locomotion based on reference hip-x joint signal acquired from cat walking. In this neural oscillator optimization one bacterium has 6 genes denoted by $G^{(n)}$ representing the neural oscillator parameters. $G_1^{(n)}$, $G_2^{(n)}$, and $G_3^{(n)}$ represent $\tau$, $\tau'$, and $\tau_f$, respectively. The other genes $G_4^{(n)}$, $G_5^{(n)}$, $G_6^{(n)}$ represent the synaptic weights tabulated in Table 5.9.

**Table 5.8:** *Parameter Values of Optimization*

| Params | $N_{gen}$ | $N_{bac}$ | $N_{clone}$ | $\alpha$ | $\beta$ | $b$ | $t_1$ | $t_2$ | $t_3$ |
|---|---|---|---|---|---|---|---|---|---|
| Value | 400 | 100 | 20 | 0.005 | 1.0 | 2.5 | 24 | 49 | 74 |

BMA parameters and the result of optimization are tabulated in Tables 7.1 and 5.9, respectively, where in Table 7.1, $N_{gen}$, $N_{bac}$, and $N_{clone}$ are the number of generations, number of bacteria, and the number of clones respectively, $b$ is the rate of the adaptation value of neural oscillator. We set parameter $t_1 - t_3$ based on the preliminary test. The neural oscillator signal is depicted in Fig. 5.30. Since the walking cat trajectory has $\pi/2$ phase difference in one walking cycle, the neuron oscillator also has $\pi/2$ phase difference as designed in the optimization evaluation. This phase difference can be changed depending on the feedback sensor from sensory neurons.

**Table 5.9:** *Representation Parameters and Results*

| Gene Params. | $G_1^{(n)}$ | $G_2^{(n)}$ | $G_3^{(n)}$ | $G_4^{(n)}$ | $G_5^{(n)}$ | $G_6^{(n)}$ |
|---|---|---|---|---|---|---|
| Neural params. | $\tau$ | $\tau'$ | $\tau_f$ | $w_{0,1}$; $w_{1,2}$; $w_{2,3}$; $w_{3,0}$ | $w_{1,0}$; $w_{0,3}$; $w_{3,2}$; $w_{2,1}$ | $w_{0,2}$; $w_{2,0}$; $w_{1,3}$; $w_{3,1}$ |
| Value | 2.0 | 12.0 | 7.05 | 2.431 | 0.789 | 2.248 |

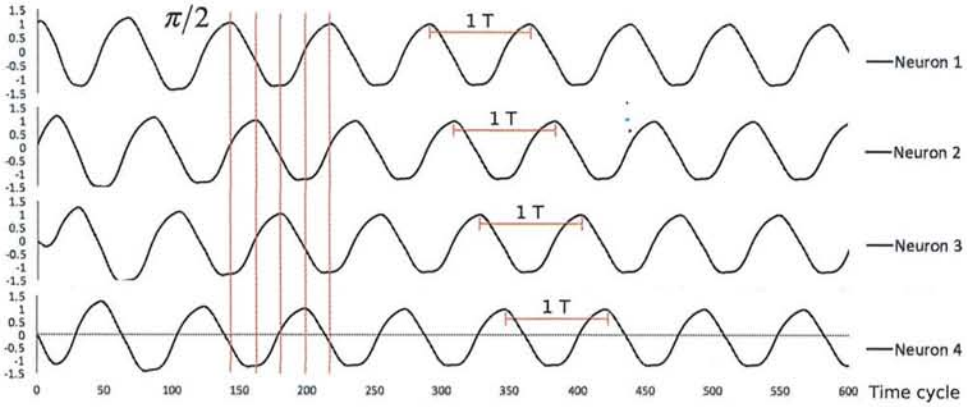



**Figure 5.30:** *Neural oscillator signal as the input of the passive neural system*

### 5.3.4.2 Bezier Curve Optimization

After optimizing the neuron structure, we optimize the walking model. This walking model requires 3 optimization processes in each leg which represents the relationship of neural signal and joint trajectories (hip-y, hip-x, and knee). There are 12 points that are required to be optimized for one optimization process. We use the same optimization parameters presented in Table 7.1 as in the previous optimization process.

The evolutionary algorithm successfully decreased the error as illustrated in Fig. 5.31. The evolution of the Bezier curve model in certain generations can be seen in Fig. 5.32. The comparison graph between the reference and the optimized result is shown in Fig. 5.33, where neural signal is shown as x-axis and joint angle value as y-axis. Since 3 joints have to be optimized, thus 3 graphs are shown. Based on the result in Fig. 5.33, the Bezier based optimization successfully forms the graph similar to the reference graphics.

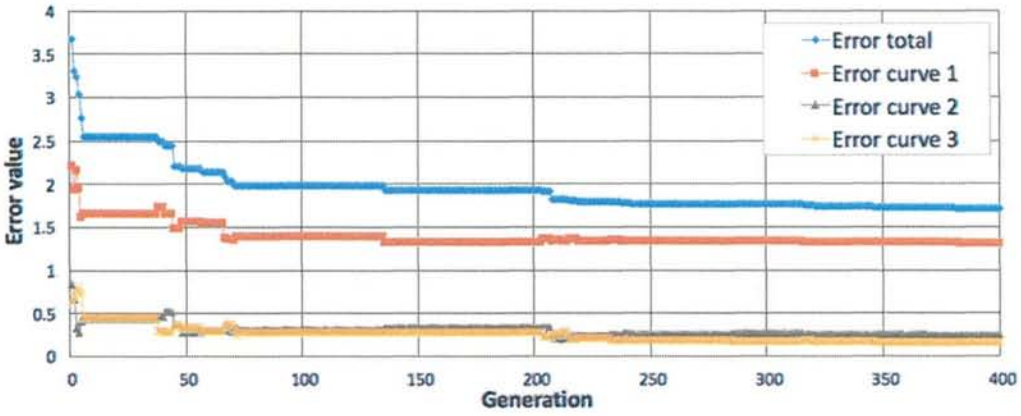



**Figure 5.31:** *Fitness evolution*

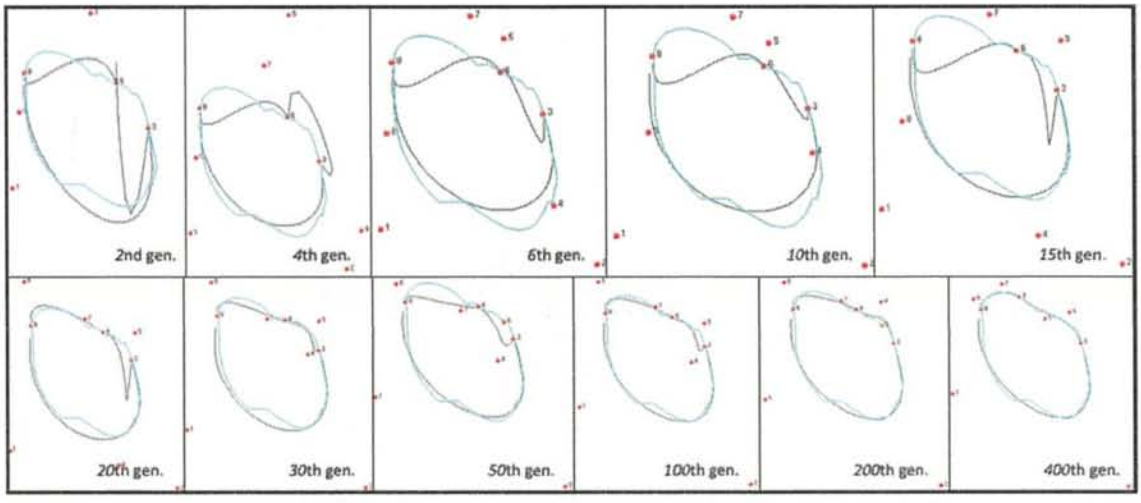



**Figure 5.32:** *A sample of Bezier curve evolution in certain generations*

**Table 5.10:** *Optimized Parameters in Bézier based Optimization*

| n th curve | Point | LEG 1 | | | | | | LEG 2 | | | | | | LEG 4 | | | | | | LEG 3 | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Hip-y | | Hip-x | | Knee | | Hip-y | | Hip-x | | Knee | | Hip-y | | Hip-x | | Knee | | Hip-y | | Hip-x | | Knee | |
| | | x | y | x | y | x | y | x | y | x | y | x | y | x | y | x | y | x | y | x | y | x | y | x | y |
| 1 | $P_0$ | -0.31 | -0.03 | -0.02 | 0.33 | -0.02 | -1.27 | 0.47 | -0.05 | 0.47 | 1.01 | 0.47 | -1.17 | -0.6 | -0.08 | -0.6 | 0.64 | -0.6 | -1.4 | -0.14 | -0.08 | -0.14 | 0.87 | -0.14 | -1.37 |
| | $P_1$ | -0.6 | 0.01 | -0.51 | 0.54 | 0.309 | -1.06 | 0.61 | -0.02 | -0.23 | 0.76 | 0.58 | -1.74 | -0.68 | 0.07 | -0.61 | 1.04 | -0.06 | -1.4 | 0.66 | 0.02 | 0.56 | 1.06 | 0.53 | -1.37 |
| | $P_2$ | -0.59 | 0.04 | -0.66 | 0.56 | 0.348 | -3.3 | 0.14 | 0.05 | -0.69 | 0.93 | 0.45 | -2.86 | 0.43 | 0.1 | 0.16 | 0.72 | 0.26 | -1.31 | 0.55 | 0.11 | 0.55 | 0.87 | 0.49 | -0.82 |
| | $P_3$ | -0.59 | 0.04 | -0.57 | 0.77 | 0.489 | -1.69 | -0.1 | 0.05 | -0.58 | 0.55 | 0.3 | -1.91 | 0.51 | 0.02 | 0.49 | 1.07 | 0.48 | -1.18 | 0.38 | 0.11 | 0.47 | 1.47 | 0.5 | -1.57 |
| 2 | $P_0$ | -0.63 | 0.11 | -0.57 | 0.77 | 0.489 | -1.69 | -0.1 | 0.05 | -0.58 | 0.55 | 0.3 | -1.91 | 0.51 | 0.02 | 0.49 | 1.07 | 0.48 | -1.18 | 0.38 | 0.11 | 0.47 | 1.47 | 0.5 | -1.57 |
| | $P_1$ | 0.46 | -0.01 | 0.43 | 1.07 | 0.51 | -0.8 | -0.63 | 0.09 | -0.05 | 0.46 | 0.13 | -0.94 | 0.42 | -0.07 | 0.51 | 2.25 | 0.55 | -1.91 | 0.12 | 0.03 | 0.38 | 1.49 | 0.38 | -3.38 |
| | $P_2$ | 0.5 | -0.09 | 0.599 | 0.79 | 0.519 | -1.33 | -0.59 | 0.04 | -0.17 | 0.18 | 0.06 | -1.33 | 0.3 | -0.06 | 0.39 | 0.19 | 0.46 | -2.75 | 0.07 | 0.06 | 0.34 | 0.32 | 0.3 | -1.13 |
| | $P_3$ | 0.5 | -0.09 | 0.46 | 1.45 | 0.04 | -1.35 | -0.59 | 0.01 | 0.19 | 0.37 | -0.6 | -1.4 | -0.04 | -0.08 | -0.04 | 0.3 | 0.26 | -1.93 | -0.14 | 0.02 | 0.11 | 0.3 | 0.01 | -1.26 |
| 3 | $P_0$ | -0.44 | -0.3 | 0.46 | 1.45 | 0.04 | -1.35 | -0.59 | 0.01 | 0.19 | 0.37 | -0.6 | -1.4 | -0.04 | -0.08 | -0.04 | 0.3 | 0.26 | -1.93 | -0.14 | 0.02 | 0.11 | 0.3 | 0.01 | -1.26 |
| | $P_1$ | 0.25 | -0.05 | 0.359 | 1.38 | -1.15 | -1.46 | -0.57 | -0.09 | 0.39 | 0.57 | -0.13 | -1.38 | -0.48 | -0.12 | -0.45 | 0.61 | 0.03 | -0.98 | -0.83 | -0.08 | -0.74 | 0.53 | -0.13 | -1.35 |
| | $P_2$ | -0.02 | -0.08 | 0.358 | 0.09 | -0.35 | -1.39 | 0.53 | -0.14 | 0.55 | 2.21 | 0.34 | -1.32 | -0.54 | -0.09 | -0.58 | 0.43 | 0.17 | -1.26 | -0.68 | -0.13 | -0.86 | 0.76 | -1.2 | -1.45 |
| | $P_3$ | 0.33 | -0.08 | -0.02 | 0.33 | -0.02 | -1.27 | 0.47 | -0.05 | 0.47 | 1.01 | 0.47 | -1.17 | -0.6 | -0.08 | -0.6 | 0.64 | -0.6 | -1.4 | -0.14 | -0.08 | -0.14 | 0.87 | -0.14 | -1.37 |

After the relationship graphs are optimized, the inversion process is required in order to invert $t$ parameter as output in the Bezier equation for x-axis. Therefore, by giving the neuron signal value, we get the $t$ parameter, then the $t$ value is used as input in Bezier equation for y-axis. Indirectly the output of angle joint is acquired from the value of motor neuron.

After inversion process, the comparison graphs between joint references and the output joint of Bezier model were recorded. In Fig. 5.34 x-axis is the time cycle and y-axis is the joint angle values. The representation of neuron ID to leg ID is decided based on preliminary analysis. The output of hip-y, hip-x, and knee joint is generated based on the input of neuron as the signal generator. In time based graph, the joints comparison between the reference and the output of the proposed system is not much different. Only some small ripples are occurring, such as in the knee output joint in second, third, and fourth leg. These ripples do not affect the performance of the locomotion.

In order to prove the effectiveness of the proposed model, we applied the proposed system in a small 4 legged robot which has only 16 MHz microcontroller. This neuro-locomotion is successfully implemented in the robot. We run the robot through rough terrain, in the sloped terrain with $8°$ and $16°$ slope which are shown in Fig. 7.18. The recorded oscillation body tilt robot in pitch and roll direction can be seen in Fig.5.36 and the effect of ground sensor can be seen in Fig. 5.37. In the computational cost evaluation, this proposed system can be significantly reduced by 63% from the previous neuro-locomotion design [178] from $9.744 \cdot 10^{-3}$ seconds to $3.603 \cdot 10^{-3}$ seconds for 1000 times looping. Therefore, this proposed locomotion model is able to be applied to low cost 4 legged robot locomotion.

### 5.3.5 Discussion

The sensor system integration is required in order to realize the dynamic locomotion. The influence of the feedback sensor to the locomotion system is designed based on the preliminary studies. We used simple sensor integration between body tilt angle, ground sensor, and the output of joint angle. In future, advance and complex sensor integration is required in order to reach better performance. It is important in neuro-based locomotion in order to generate the dynamic signal and to enable walking on unstructured terrain. However the complexity of the sensor integration will increase the computational cost. Timing control of neural signal for responding sensor feedback will be implemented. Parameter $g_i$ in Eq. (5.68) is required in order to fit into the relationship graph based on Bezier model. The proposed passive neural control based neuro-locomotion can be extended to 3-dimensional Bezier model, such that the signal from ground and tilt sensor can cause the changes of neural signal and can affect the joint trajectory as well.

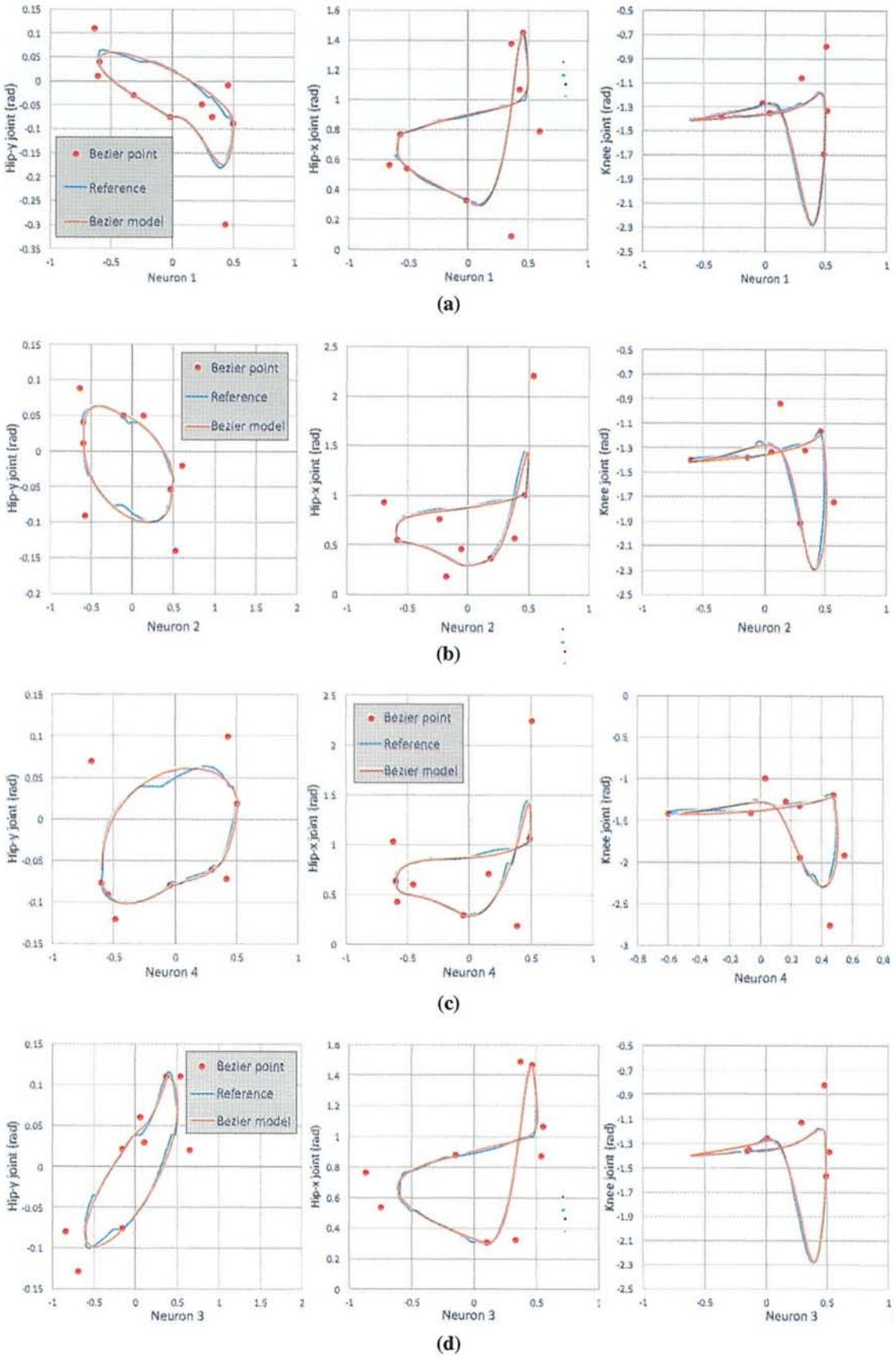



**Figure 5.33:** *Reference and Bezier output of relationship between neuron signal and joint trajectories (hip-y, hip-x, and knee) (a) first leg generated by Neuron 1 (b) second leg generated by Neuron 2 (c) third leg generated by Neuron 4 (d) fourth leg generated by Neuron 3*

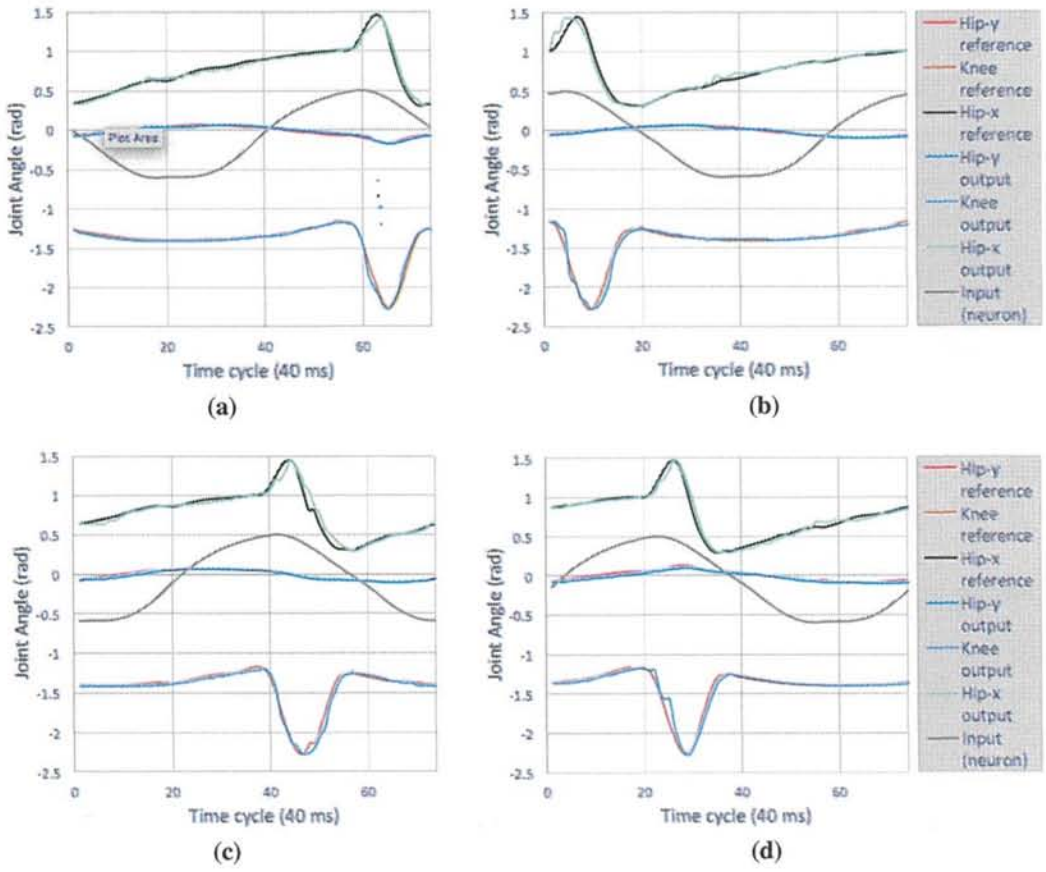



**Figure 5.34:** *Comparison between joint reference and Bezier output of angle joint extracted during robot walking with neural signal as the input generator in time cycle based graphic (a) first leg generated by Neuron 1 (b) second leg generated by Neuron 2 (c) third leg generated by Neuron 4 (d) fourth leg generated by Neuron 3*

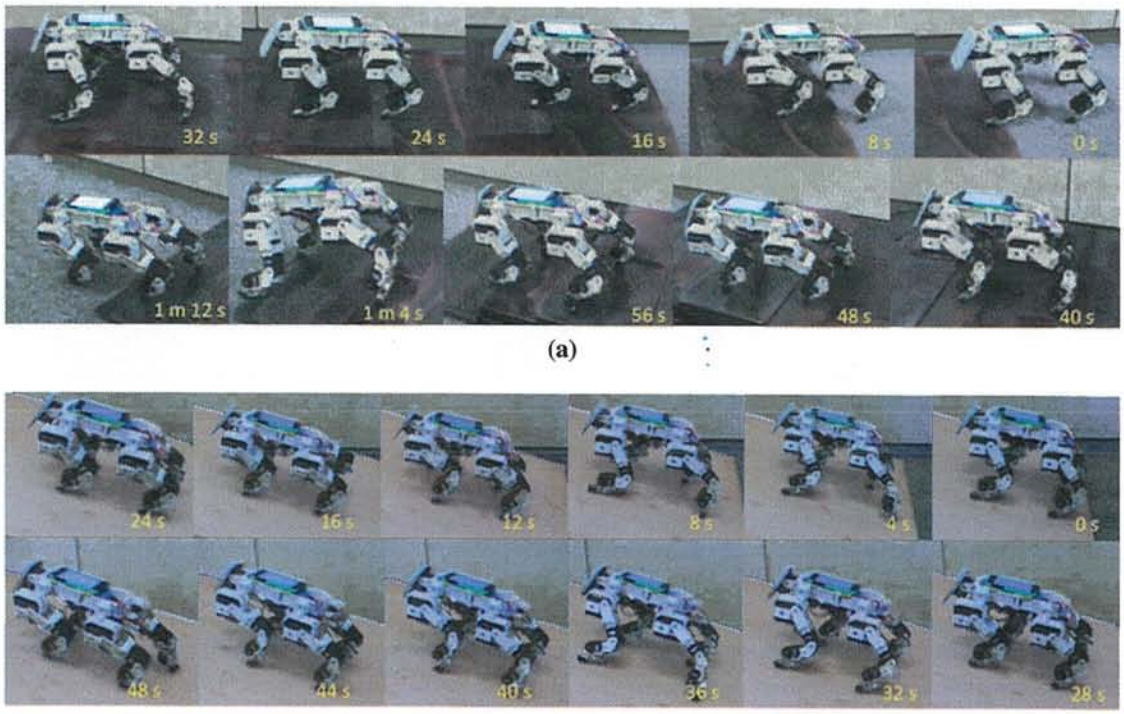



**Figure 5.35:** *Four legged robot walking experiment (a) rough terrain (b) slope terrain with* 16° *slope*

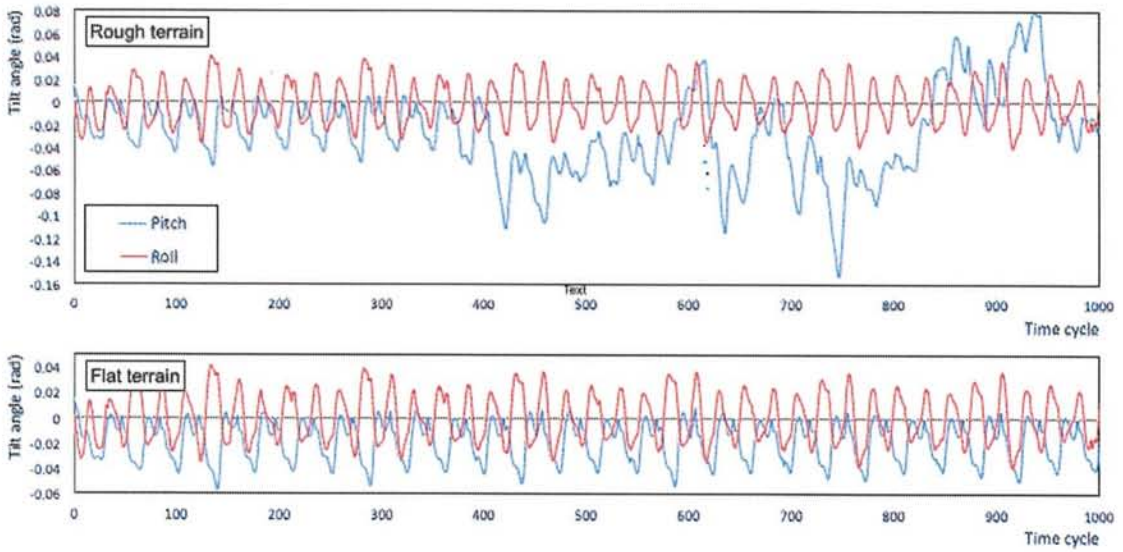



**Figure 5.36:** *Body's tilt recorded during robot walking in rough terrain and flat terrain shows the stability can be reached by using the proposed model. In rough terrain experiment, the body tilt angle in pitch direction has big oscillation because the robot walking is uphill and downhill on the rough terrain.*

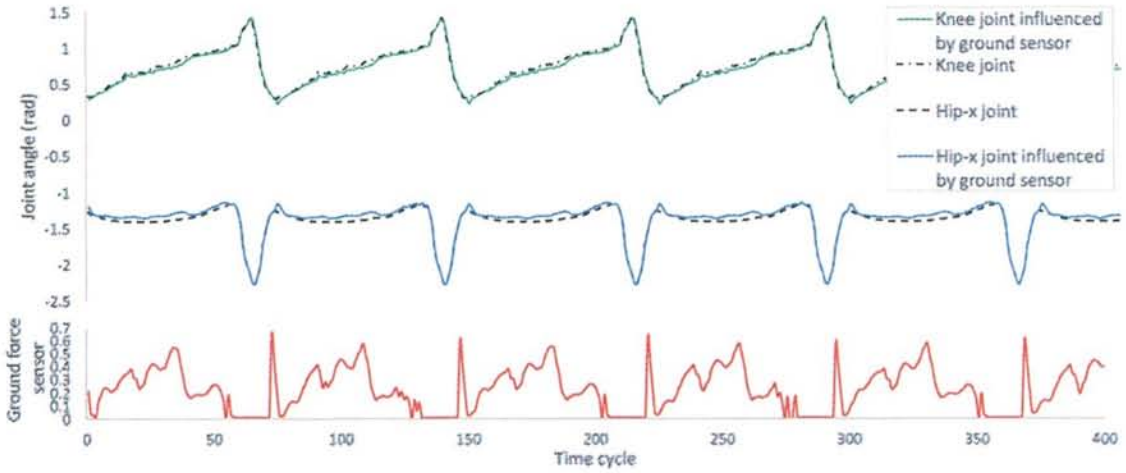



**Figure 5.37:** *The effect of ground sensor. The data are taken from first leg during robot walking. The graphic shows the comparison between the joint angle without influenced by ground sensor and the joint angle influenced by ground sensor.*

In optimization based on Bezier curve model, timing parameter is still used for relating x-axis data and y-axis data. In this proposed system, based on the preliminary test, we decided the number of curves representing the reference of optimized graph. The complexity of the graph implies the number of Bezier curves.

This section proposed a new model of neuro-locomotion by implementing passive neural control represented by Bezier curve model. There are 4 neurons in the neural structure where one neuron represents 3 joints in one leg. Recorded joint trajectory of cat-like robot is required as reference of joints. In order to optimize the relationship between signal generated by neuron and each joint, Bezier based optimization is implemented which has successfully established the relationship equation to generate the angle joint values with neural signal as input. Neural oscillator is optimized by evolutionary algorithm and is able to generate signal with $\pi/2$ phase difference that is suitable for walking pattern of cat. Phase difference can be different depending on the feedback signal from sensory neuron. This proposed system can reduce the neural complexity and also significantly reduce the computational cost by 63% as well. Therefore, this proposed neuro-locomotion system has successfully been applied to the small 4 legged robot with low frequency microcontroller and considers the influence of sensory feedback.

## 5.4 Walking Speed Control Behavior

This section proposes a gait generation system based on human behavior using biological approach. Humans have different gaits with different levels of speed or step. The proposed

gait generator is able to generate the walking transition when the speed and the step length are changing dynamically. Neuron interconnection structures as the locomotion model are formed. We apply evolutionary computation for each level of walking optimization.

### 5.4.1 Introduction

The development of robot locomotion is increasing significantly. Trajectory-based models are the most famous models for humanoid locomotion. Trajectory-based locomotion concept is processed from outside (locomotion pattern) to inside (locomotion generator). The locomotion generator parameter is generated based on the defined trajectory pattern. Many researchers have implemented this in their humanoid robot. We used polynomial equation in order to form the trajectory pattern [176, 177], [237]. Some researchers used zero moment point (ZMP) model for stability support and other researchers implemented inverted pendulum in humanoid robot locomotion. Chevallereau et al. also implemented the extended model of inverted pendulum for self-stabilization in their humanoid robot [33].

In contrast to trajectory-based locomotion, human behavior inspired locomotion is processed from inside (locomotion generator) to outside (locomotion pattern). This locomotion is inspired by human behavior analysis investigating how humans learn their walking model and automatically separate it into different walking patterns with different speeds and step lengths of walking. This model implements a biological approach for locomotion model and gait generator.

Locomotion control using a biological approach is not easier than trajectory-based locomotion. Several researchers have proposed the locomotion based on biological approaches. Most of these have implemented coupled neuron oscillators for generating the locomotion pattern. Matos et al. implemented central pattern generation (CPG) in their bipedal robot locomotion [127]. CPG is also implemented for biomimetic models to control turning motion of a snake-like robot [144]. Iwasaki also implemented neuronal circuits for controlling walking patterns without controlling the walking speed [89]. Another neuro-locomotion model based on multi-layered neuron structure was proposed by Nassour et al. for bipedal robot locomotion. Their proposed locomotion model achieved a satisfactory walking pattern with good stability, however, they did not consider walking speed and step length control [142]. Ishiguro et al. proposed neural oscillator-based biped robot locomotion. Feedback sensors were applied in order to realize dynamic walking. This system did not consider walking speed control [88]. Other researchers combined central pattern generation with evolutionary algorithms. In order to acquire good walking pattern, Park et al. implemented evolutionary algorithm to optimize the synaptic weight value in the neuron structure of CPG [148]. Bay-

din also implemented single objective evolutionary algorithm to optimize CPG parameter in his locomotion model. He realized 2-D locomotion that did not consider the disturbance effect [22]. Inada et al., Hong et al., and Chernova et al. also applied evolutionary algorithm in their bipedal robot locomotion models [87], [149], [31]. The major problems in biologically-inspired locomotion are stability and walking speed or behavior control. The aim of the proposed research is to control the gait generator in order to be able to generate the walking transition with dynamically changing speed and step length.

In previous research, we developed the locomotion that implemented a biological approach. The locomotion could perform only one walking pattern, therefore, we could not control the walking speed and walking step. The previous locomotion model used fix interconnection model formed based on the human body analysis and some preliminary tests. The variant walking were generated by changing the gain signal which effected the amplitude of neuron signal in joint angle level [168]. In this research, we developed the gait generator for generating different walking patterns with different levels of speed. We form the interconnection structure of neural oscillator in several different levels of speed and step length for optimizing suitable walking pattern. For optimizing the interconnection structure and its synaptic weights, we apply an evolutionary computation technique, the Bacterial Memetic Algorithm (BMA) [25]. The gait generator system acquires the dynamic relationship between walking speed and walking patterns by using MLP with back-propagation algorithm and uses the optimized MLP parameters for generating the parameters of the locomotion generator.

One contribution of the proposed system is, that we can generate different walking patterns with different levels of speed and also form their walking transitions by using biological approach, while other researchers who used biological approach did not consider different walking patterns' generation. BMA is used for optimizing the biological based robot locomotion.

### 5.4.2 Locomotion System

In this section the gait generator model for this proposed locomotion is explained. The locomotion system is divided into three subsystems depicted in Fig. 7.1. The walking model provides the optimized walking patterns $(P_1, P_2, P_3, \cdots, P_{20})$ with different walking speeds. In the gait generator system, combining twenty walking patterns is conducted by using MLP with back-propagation algorithm. After the MLP's weight parameters are optimized, the gait generator is ready to generate the walking parameters $(G_1, G_2, G_3, \cdots, G_{30})$ with walking speed input $(d_v)$ by using a feed-forward MLP with optimized parameters. The walking

parameters will be converted to synaptic weights of neuro-locomotion and also converted to joint gains in the locomotion generator system.

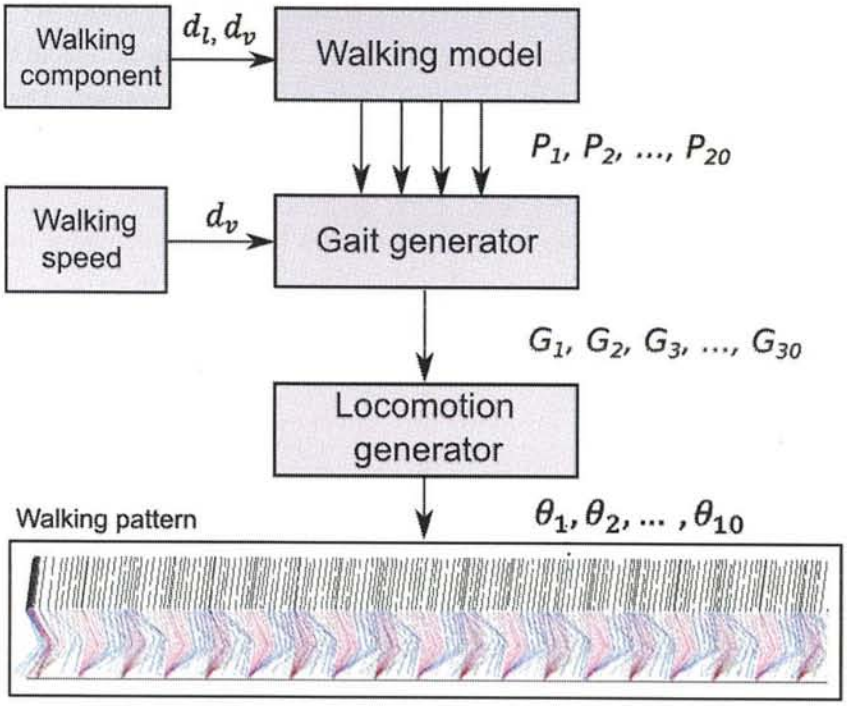



**Figure 5.38:** *Diagram of the locomotion system*

Algorithm 5.1 shows the process of the proposed locomotion model. The first process is the walking pattern formation process. After that, gait generator learning system is processed. In gait generator, after the parameters of MLP with back-propagation algorithm are optimized, the robot will be ready for a walking command.

#### 5.4.2.1 Walking Model

In this section, the locomotion generator and the strategy for synaptic connectivity optimization are explained. The detailed synaptic connectivity optimization for neuro-locomotion was explained in our previous research [175]. We conducted the synaptic connectivity optimization by using evolutionary computation twenty times with different fitness parameters for resulting twenty walking patterns with different walking speeds and walking step lengths. The walking pattern optimization process can be seen in Algorithm 5.2. In the first step, we decided the desired walking speed parameter and run the BMA evolutionary optimization technique with around 300000 evaluations. After the walking pattern parameters were acquired, we kept the optimized walking pattern ($P_1$) and repeated the evolutionary

**Algorithm 5.1** Locomotion system

Input: A walking speed and a walking step length
Output: Joint angle values
Data initialization
Walking pattern formation
Gait generator learning system
Robot is ready for walking
**while** Locomotion system is running **do**
  ReadDesiredWalking($d_v$)
  **if** $d_v$ *is changed* **then**
    GaitGenerator($d_v$)
    Resulting walking pattern $(G_1, G_2, G_3, \cdots, G_{30})$
  **end if**
  LocomotionGenerator$(G_1, G_2, G_3, \cdots, G_{30})$
  Resulting joint angle values $(\theta_1, \theta_2, \cdots, \theta_{10})$
**end while**

process with different desired walking speed until the 20th walking pattern parameters ($P_{20}$) were acquired.

**Algorithm 5.2** Walking pattern formation

Input: Speed, step length, tilt angle, and angular velocity of the robot (Walking Evaluation Component)
Output: Walking pattern parameters
Forming walking patterns
**for** $i \leftarrow 1$ to 20 **do**
  Set desired walking evaluation component
  BMA optimization process
  Resulting $i$th walking pattern parameters
**end for**
Transferring 20 walking patterns

**5.4.2.1.1 Locomotion Generator** The design of humanoid robot in this proposed research uses only leg actuators having ten degrees of freedom (DoF) for implementing walking pattern without stability system. We implemented a coupled neural oscillator-based locomotion generator. This coupled neural oscillator represents only the important joints for forming the walking pattern, which are the knee joint and hip-x joint as depicted in Fig. 5.39. One joint angle is represented by a couple motor neuron with reciprocal inhibition. The neural oscillator model for this locomotion was presented in the previous research [178], [130]. The neural locomotion model in this proposed research is illustrated in Fig. 5.39, and its

mathematical model is shown in Eqs. (5.65), (5.66), (5.67).

$$\tau \dot{u}_i = (u_0 - u_i - \sum_{j=1}^{n} w_{ij} y_j - b v_i)\tau_f \tag{5.47}$$

$$\tau' \dot{v}_i = (y_i - v_i)\tau_f \tag{5.48}$$

$$y_i = \max(u_i, 0) \tag{5.49}$$

$$\theta_i^{(l,r)} = (y_i - y_{i+1})G_{28+i} \tag{5.50}$$

where $u_i$, $y_i$, and $v_i$ are the inner state, output value, and a variable that represents the adaptation value or the self-inhibition effect of the $i$th neuron, respectively; $b$ is the rate of the adaptation value. The external input for coupled neurons, which has a constant rate, is denoted by $u_0$. The time constant of the inner state and the adaptation effect in the neuron are represented by $\tau$ and $\tau'$, respectively. In Eq. (5.65), $w_{ij}$ represents the strength of the inhibitory effect between the motor neurons that is optimized offline; $\sum_{j=1}^{n} w_{ij} y_j$ represents the total of the signal input from the neuron. In Eqs. (5.65) and (5.66), $\tau_f$ is used for controlling the frequency of oscillation.

In this proposed research we considered the walking pattern with different walking speeds implying that only hip-x and knee joint influence those walking patterns.

Locomotion generator converts the walking parameters $(G_1, G_2, \cdots, G_{30})$ from both the individuals in evolutionary process and gait generator process into synaptic weights of neuro-locomotion and their gain values. Those parameters represent the hip-x joint and knee joint ($\theta_1^{(l,r)}$ and $\theta_2^{(l,r)}$) computed in Eq. (5.68). In the walking pattern formation system, the locomotion generator system generated the locomotion to acquire the fitness calculation for the evolutionary optimization process. The other joints which are not represented by walking pattern parameters are simple computed by Eqs. (5.51), (5.52), and (5.53).

$$\theta_0^{(l,r)} = \theta_1^{(l)} - \theta_1^{(r)} \tag{5.51}$$

$$\theta_3^{(l,r)} = \theta_2^{(l,r)} - \theta_1^{(l,r)} \tag{5.52}$$

$$\theta_4^{(l,r)} = -\theta_0^{(l,r)} \tag{5.53}$$

Where $\theta_0^{(l,r)}$, $\theta_3^{(l,r)}$, and $\theta_4^{(l,r)}$ represent the angle value of hip-y joint, ankle-x joint, and ankle-y joint in left or right leg, respectively. In Eq. (5.51), $\theta_1^{(l)}$ and $\theta_1^{(r)}$ are the angle value of hip-x joint in left and right position, respectively.

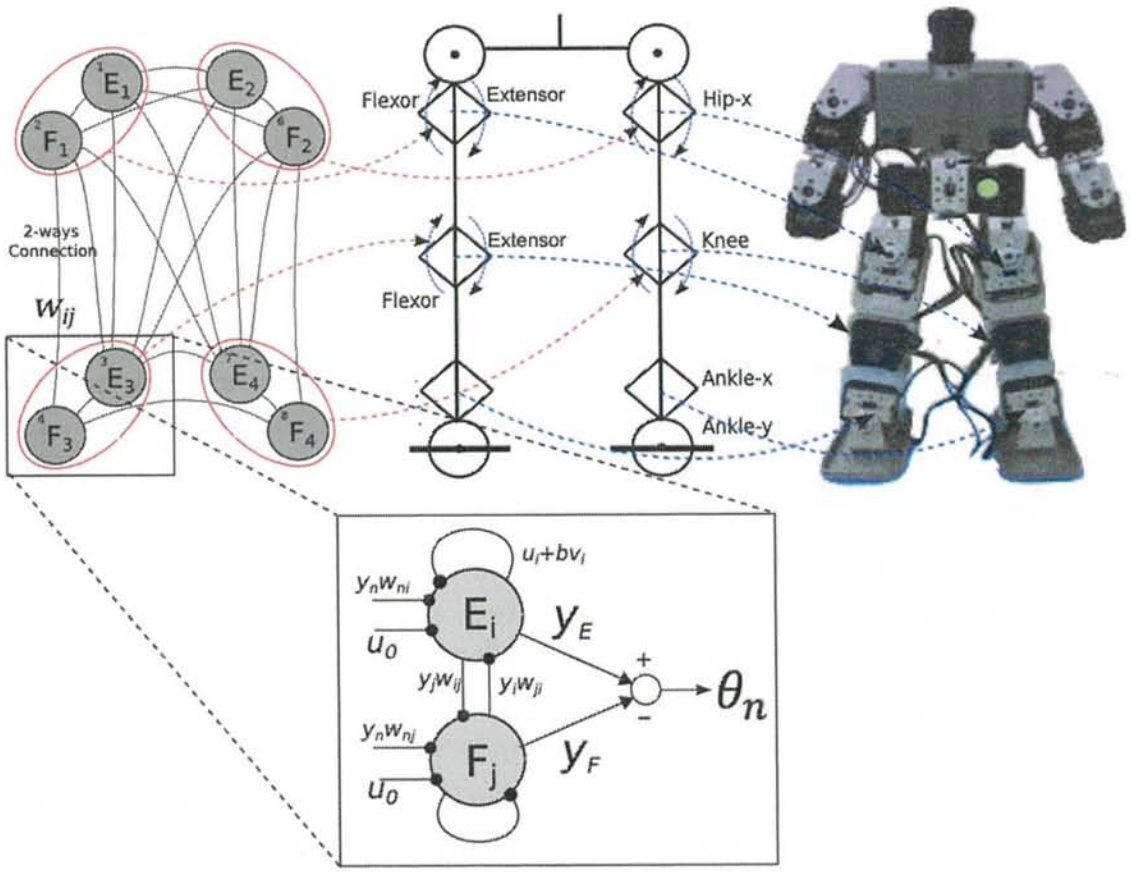



**Figure 5.39:** *Neuro-locomotion structure*

**5.4.2.1.2 Synaptic Connectivity Optimization** For finding the optimal solutions for the problems discussed in this research the bacterial memetic algorithm is applied [25]. BMA is a population based stochastic optimization technique which effectively combines global and local search in order to find good quasi-optimal solution for the given problem. In the global search, BMA applies the bacterial operators (bacterial mutation and gene transfer). The role of the bacterial mutation is the optimization of the bacteria's chromosome. The gene transfer allows the information's transfer in the population among the different bacteria. Levenberg-Marquardt method is applied as the local search technique.

The operation of the BMA starts with the generation of a random initial population containing $N_{ind}$ individuals. Next, until a stopping criterion is fulfilled (which is usually the number of generations, $N_{gen}$), we apply the bacterial mutation, local search, and gene transfer operators. Bacterial mutation creates $N_{clones}$ number of clones (copies) of an individual, which are then subjected to random changes in their genes. The number of genes that are modified with this mutation is a parameter of the algorithm ($l_{bm}$). In the local search step for each individual the Levenberg-Marquardt algorithm is used by a certain probability till a certain number of iterations. The two other parameters of this step is the initial bravery factor ($\gamma$ init.) and the parameter for the terminal condition ($\tau$). The last operator in a generation is the horizontal gene transfer. It means copying genes from better individuals to worse ones. For this reason, the population is split into two halves, according to the fitness values. The number of gene transfers in one generation ($N_{inf}$), as well as the number of genes ($l_{gt}$), that get transferred with each operation, are determined by the parameters of the algorithm.

Bacterial memetic algorithm has been successfully applied to wide range of problems. More details about the algorithm can be found in [25], [18].

In case of evolutionary and memetic algorithms we have to discuss the encoding method and the evaluation of the individuals (bacteria) as well. The synaptic connectivity optimization was developed in the previous research, where multi-objective evolutionary algorithm was used for optimizing the synaptic weight values in the neural locomotion [175]. In this proposed research, weight factors are used for calculating the fitness values in the multi-objective problem as a single-objective problem. Speed walking changing, oscillation of tilt angle, and step length appropriate to the desired length of step are the objectives in this optimization model. We use two-dimensional walking simulation for calculating the objective data. Four objectives are to be minimized to acquire good walking pattern. In the BMA, the bacterium contains the walking pattern parameters described by 30 real numbers which correspond to the 30 genes ($x_1 - x_{30}$). Those genes ($x_1 - x_{30}$) are converted into 2 parameters computed in Eqs. (5.69) and (5.70); 28 genes ($G_1, G_2, G_3, \cdots, G_{28}$) represent the weight values and the neuron structure and the remaining 2 genes ($G_{29}, G_{30}$) represent the

**Table 5.11:** *Interconnection Maps Represented by Genes*

| | $W_{(j)1}$ | $W_{(j)2}$ | $W_{(j)3}$ | $W_{(j)4}$ | $W_{(j)5}$ | $W_{(j)6}$ | $W_{(j)7}$ | $W_{(j)8}$ |
|---|---|---|---|---|---|---|---|---|
| $W_{1(i)}$ | 0 | $G_1$ | $G_2$ | $G_3$ | $G_4$ | $G_5$ | $G_6$ | $G_7$ |
| $W_{2(i)}$ | $G_8$ | 0 | $G_9$ | $G_{10}$ | $G_{11}$ | $G_{12}$ | $G_{13}$ | $G_{14}$ |
| $W_{3(i)}$ | $G_{15}$ | $G_{16}$ | 0 | $G_{17}$ | $G_{18}$ | $G_{19}$ | $G_{20}$ | $G_{21}$ |
| $W_{4(i)}$ | $G_{22}$ | $G_{23}$ | $G_{24}$ | 0 | $G_{25}$ | $G_{26}$ | $G_{27}$ | $G_{28}$ |
| $W_{5(i)}$ | $G_4$ | $G_5$ | $G_6$ | $G_7$ | 0 | $G_1$ | $G_2$ | $G_3$ |
| $W_{6(i)}$ | $G_{11}$ | $G_{12}$ | $G_{13}$ | $G_{14}$ | $G_8$ | 0 | $G_9$ | $G_{10}$ |
| $W_{7(i)}$ | $G_{18}$ | $G_{19}$ | $G_{20}$ | $G_{21}$ | $G_{15}$ | $G_{16}$ | 0 | $G_{17}$ |
| $W_{8(i)}$ | $G_{25}$ | $G_{26}$ | $G_{27}$ | $G_{28}$ | $G_{22}$ | $G_{23}$ | $G_{24}$ | 0 |

gain values of the joint angle in hip-x and knee. One gene represents one parameter in the neural locomotion. The genes representing the weight values and the gain values of the joint angles have different minimum values $x_n^{min}$ and maximum values $x_n^{max}$. The minimum and maximum values were decided according to the previous analysis. The genes representing the interconnection weights from origin neuron to destination are shown in Table 5.14. For example, the weight connection from neuron 1 to neuron 8 ($W_{1,8}$) is represented by $G_7$, where $G_7 = x_7(x_7^{max} - x_7^{min}) + x_7^{min}$.

$$G_j = \begin{cases} G'_j & \text{if } 1 \leq j \leq 28 \ \& \ x_j > 0 \\ G'_j & \text{if } 29 \leq j \leq 30 \\ 0 & \text{otherwise} \end{cases} \tag{5.54}$$

$$G'_j = x_j(x_j^{max} - x_j^{min}) + x_j^{min} \tag{5.55}$$

where $x_j$ is the $j$th gene of the solution in normalized state and $G_j$ is the $j$th gene that is ready to be used in the neural locomotion. For weight values, if $G_j$ is lower than zero, then, $G_j$ for synaptic weight is zero implying that there is no connection between the neurons.

**5.4.2.1.3 Calculation** In the fitness function, we minimize the error of walking speed ($E^{(v)}$), the error of desired step length ($E^{(l)}$), height of step ($h$), and the Center of Mass (CoM) error in x-axis ($E_x^{(CoM)}$). Each component of the evaluation function has a weight factor ($w_i^{(f)}$).

$$E^{(v)} = \sum_{j=1}^{T} \|d_v - v(t)\| \tag{5.56}$$

$$E^{(l)} = \|d_l - l\| \tag{5.57}$$

$$h = \begin{cases} y(t) & \text{if } h < y(t) \ \& \ x_p(t) < x_a(t) \\ h & \text{otherwise} \end{cases} \tag{5.58}$$

$$E_x^{(CoM)} = \sum_{j=1}^{T} \|CoM_x(t)\| \tag{5.59}$$

$$f = w_0^{(f)} E^{(v)} + w_1^{(f)} E^{(l)} - w_2^{(f)} h + w_4^{(f)} E_x^{(CoM)} \tag{5.60}$$

where $T$ is the maximum time and $t$ is the current time. In Eq. (5.56), $d_v$ is the desired speed, $v(t)$ is the current speed. In Eq. (5.57), $d_l$ is the desired length of step and $l$ is the length of step. In Eq. (5.58), $y(t)$ is the current height of step, $x_p(t)$ is the current pelvis position in x-axis, and $x_a(t)$ is the current ankle position in x-axis. In Eq. (5.59), $CoM_x(t)$ is the length of CoM from supported area in x-axis.

#### 5.4.2.2 Gait Generator

The gait generator uses MLP with back-propagation learning mechanism to acquire the relationship between the input of the walking pattern (walking speed) and the synaptic weight of neuro-locomotion. MLP can easily acquire the dynamic relationship between the walking speed and the synaptic weight of locomotion obtained from the walking patterns formation system.

The MLP system has one neuron in the input layer, certain number of neurons in the hidden layers, and 30 neurons in the output layer as illustrated in Fig. 5.40. The number of hidden layers can be more than one.

The gait generator process is divided into 2 processes. The first process optimizes the MLP's weight parameters to acquire the dynamic relationship between the walking speed and the synaptic weight of locomotion. After the MLP's weight parameters are optimized, the gait generator is ready for generating the synaptic weight parameters for neural locomotion with any walking speed and step length. The process of gait generator is explained in Algorithm 5.3.

In Eqs. (6.27), $x_j^{(i)}$ and $x_j'^{(i)}$ are the output value and the input value of the $j$th neuron in the

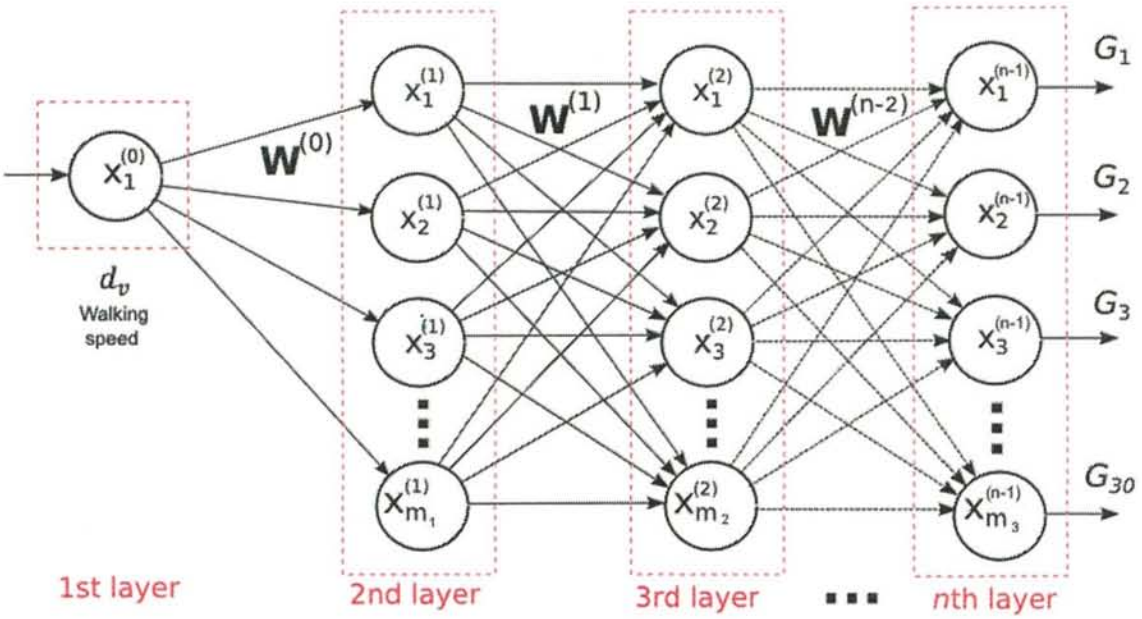



**Figure 5.40:** *Diagram of MLP with back-propagation model*

**Algorithm 5.3** Gait generator learning system

Input: Walking patterns
Output: Optimized weight parameters of MLP
**if** **W** has been optimized **then**
  MLP with walking speed input ($d_v$)
**else**
  **for** $i \leftarrow 1$ to $5 \times 10^6$ **do**
    **for** $j \leftarrow 1$ to $20$ **do**
      Train $j$th training data in $i$th MLP process
    **end for**
  **end for**
**end if**

$i$th layer, respectively; $W_{kj}^{(i-1)}$ is the weight value between the $k$th neuron in the $(i-1)$th layer and the $j$th neuron in the $i$th layer; $m[n]$ is a vector representing the number of neurons in each layer, where $n$ is the number of layers, defined as $m[4] = \{1; 10; 30; 30\}$. It represents, that there is one neuron in the 1st layer, 10 neurons in the 2nd layer, 30 neurons in the 3rd layer, and 30 neurons in the 4th layer.

$$x_j^{(i)} = f(x_j'^{(i)}) = f\left(\sum_k^{m_{i-1}} x_k^{(i-1)}(t) W_{kj}^{(i-1)}\right) \tag{5.61}$$

$$\delta_k^{(i-1)} = \begin{cases} (d_k - x_j^{(i)}) f'\left(x_j'^{(i)}\right) & \text{if } i = y \\ \sum_k^{m_i} \delta_k^{(i)} W_{jk}^{(i-1)} f'\left(x_j'^{(i)}\right) & \text{otherwise} \end{cases} \tag{5.62}$$

$$\mathbf{W}^{(i-1)}(t+1) = \mathbf{W}^{(i-1)}(t) + \eta \mathbf{x}'^{(i)}(t) \delta^{(i)} \tag{5.63}$$

In Eq. (5.62), $\delta_k^{(i-1)}$ is the error propagation in the $k$th neuron of the $i$th layer, where $d_k$ is the desired value in the $k$th neuron of the output layer. We standardize the data from 0 to 1. In Eq. (6.27), the activation function $f(x)$ is sigmoid function for each neuron. The weight parameters of neurons $W^{(i-1)}$ between the $(i-1)$th layer and the $i$th layer are computed by Eq. (5.63), where $\eta$ is the learning rate of the weights.

## 5.4.3 Experimental Results

The optimization process for this proposed locomotion should take a short time, therefore we optimized the walking pattern in computer simulation. This experiment of the proposed research is divided into three parts, such as walking patterns formation optimization as the first, gait generator learning as the second, and walking speed control as the last experiment.

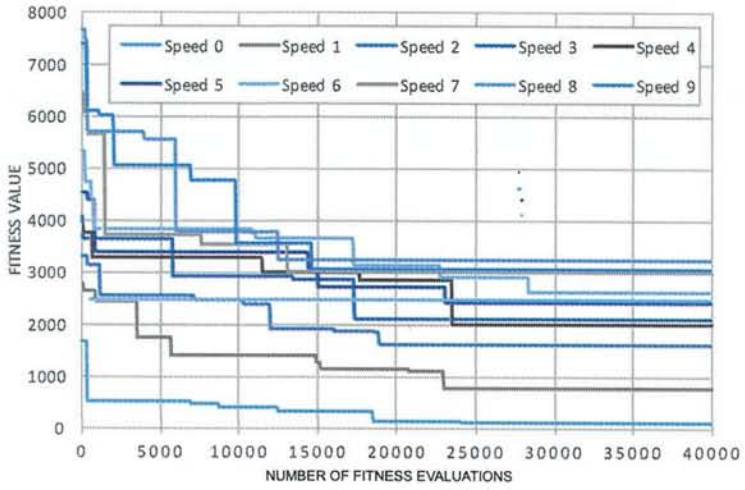
### 5.4.3.1 Walking Patterns Formation Optimization

In this experiment, we optimized the walking patterns by using BMA, whose parameters are shown in Table 5.12 representing the best parameter combination acquired from preliminary test. BMA parameters were chosen based on the result of comparison test against Steady State Genetic Algorithm (SSGA), which is a model of continuous alternation of generations, where few individuals with low fitness values are replaced by new individuals produced by genetic manipulation [195, 215]. In order to realize a fair comparison, the tests with BMA and SSGA took the same number of fitness evaluations (40000). The comparison diagram of preliminary tests between SSGA and BMA is depicted in Fig. 5.41. BMA ran 10 times (Fig. 5.41a) and SSGA ran also 10 times (Fig. 5.41b). Figure 5.41c shows the average of

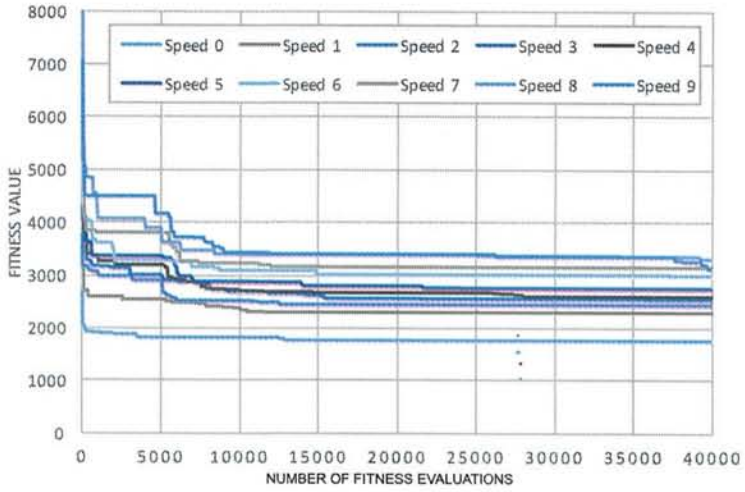
fitness evolution of BMA and SSGA. We formed 20 walking patterns with different walking speeds whose parameters are tabulated in Table 5.13 showing the desired walking speed $(d_v)$ and walking pattern parameters. Each walking speed has different walking pattern parameters.

**Table 5.12:** *BMA Parameters*

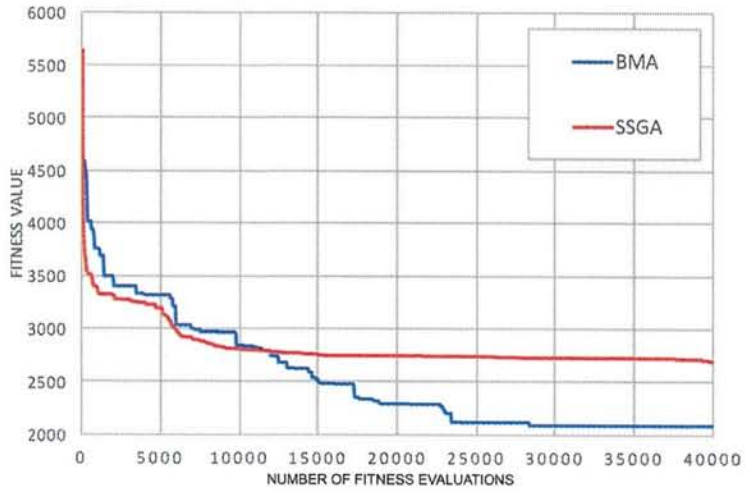
| Parameter | value |
|---|---|
| $N_{ind}$, $N_{gen}$, $N_{clones}$ | 40, 80, 4 |
| mutation segment length | 2 |
| mutation type, parameter | Gaussian, 0.3 |
| $N_{inf}$, $l_{gt}$ | 10, 10 |
| local search probability | 10 |
| maximum number of iterations | 8 |
| initial bravery factor | 1.0 |
| terminal condition | 0.0001 |
| $x^{max}_{(1-28)}$; $x^{max}_{(29-30)}$ | 3.5; 4 |
| $x^{min}_{(1-28)}$; $x^{min}_{(29-30)}$ | -3.0; 1.0 |
| $w_1^{(f)} - w_4^{(f)}$ | {0.2, 0.3, 0.3, 0.2} |




(a)




(b)




(c)

**Figure 5.41:** *Comparison diagram of fitness evolution between BMA and SSGA a) Fitness evolution of BMA b) Fitness evolution of SSGA c) Fitness average of BMA and SSGA*

**Table 5.13:** *Walking Pattern Parameters*

| | $d_x$ | $G_1$ | $G_2$ | $G_3$ | $G_4$ | $G_5$ | $G_6$ | $G_7$ | $G_8$ | $G_9$ | $G_{10}$ | $G_{11}$ | $G_{12}$ | $G_{13}$ | $G_{14}$ | $G_{15}$ | $G_{16}$ | $G_{17}$ | $G_{18}$ | $G_{19}$ | $G_{20}$ | $G_{21}$ | $G_{22}$ | $G_{23}$ | $G_{24}$ | $G_{25}$ | $G_{26}$ | $G_{27}$ | $G_{28}$ | $G_{29}$ | $G_{30}$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | . 0 - | 0 | 0 | 0 | 0.487 | 0 | 0.037 | 0 | 0.511 | 0 | 0 | 0 | 0 | 0 | 0.905 | 0.869 | 0 | 0.127 | 0 | 0.126 | 0 | 0 | 0.341 | 0.948 | 0 | 0.27 | 0 | 0 | 0.589 | 0.466 | 0.535 |
| 2 | 0 | 0 | 0 | 0 | 0.573 | 0 | 0.56 | 0 | 0.506 | 0 | 0 | 0 | 0 | 0 | 0.238 | 0.393 | 0 | 0.184 | 0 | 0.638 | 0 | 0 | 0.88 | 0.699 | 0 | 0.494 | 0 | 0 | 0.567 | 0.442 | 0.936 |
| 3 | 0.066 | 0 | 0 | 0 | 1 | 0 | 0.101 | 0 | 0.658 | 0 | 0 | 0 | 0 | 0 | 0.042 | 0.264 | 0 | 0.026 | 0 | 0.142 | 0 | 0 | 0.086 | 0.164 | 0 | 0.997 | 0 | 0 | 0.046 | 0.053 | 0.349 |
| 4 | 0.066 | 0 | 0 | 0 | 0.389 | 0 | 0.158 | 0 | 0.148 | 0 | 0 | 0 | 0 | 0 | 0.11 | 0.569 | 0 | 0.932 | 0 | 0.403 | 0 | 0 | 0.01 | 0.316 | 0 | 0.977 | 0 | 0 | 0.948 | 0.32 | 0.725 |
| 5 | 0.138 | 0 | 0 | 0 | 1 | 0 | 0.851 | 0 | 0.126 | 0 | 0 | 0 | 0 | 0 | 0.539 | 0.168 | 0 | 0.891 | 0 | 0.573 | 0 | 0 | 0.069 | 0.791 | 0 | 0.923 | 0 | 0 | 0.859 | 0.347 | 0.975 |
| 6 | 0.16 | 0 | 0 | 0 | 1 | 0 | 0.936 | 0 | 0.502 | 0 | 0 | 0 | 0 | 0 | 0.358 | 0.43 | 0 | 0.352 | 0 | 0.107 | 0 | 0 | 0.292 | 0.307 | 0 | 0.085 | 0 | 0 | 0.026 | 0.907 | 0.823 |
| 7 | 0.205 | 0 | 0 | 0 | 0.745 | 0 | 0.775 | 0 | 0.145 | 0 | 0 | 0 | 0 | 0 | 0.042 | 0.367 | 0 | 0.852 | 0 | 0.321 | 0 | 0 | 0.193 | 0.61 | 0 | 0.497 | 0 | 0 | 0.785 | 0.593 | 0.996 |
| 8 | 0.235 | 0 | 0 | 0 | 0.845 | 0 | 0.738 | 0 | 0.323 | 0 | 0 | 0 | 0 | 0 | 0.294 | 0.711 | 0 | 0.309 | 0 | 0.404 | 0 | 0 | 0.274 | 0.805 | 0 | 0.401 | 0 | 0 | 0.222 | 0.499 | 0.973 |
| 9 | 0.276 | 0 | 0 | 0 | 0.596 | 0 | 0.923 | 0 | 0.474 | 0 | 0 | 0 | 0 | 0 | 0.08 | 0.37 | 0 | 0.732 | 0 | 0.046 | 0 | 0 | 0.106 | 0.293 | 0 | 0.443 | 0 | 0 | 0.83 | 0.759 | 0.945 |
| 10 | 0.306 | 0 | 0 | 0 | 0.977 | 0 | 0.993 | 0 | 0.862 | 0 | 0 | 0 | 0 | 0 | 0.399 | 0.379 | 0 | 0.426 | 0 | 0.514 | 0 | 0 | 0.125 | 0.166 | 0 | 0.772 | 0 | 0 | 0.887 | 0.907 | 1 |
| 11 | 0.331 | 0 | 0 | 0 | 0.975 | 0 | 0.992 | 0 | 0.84 | 0 | 0 | 0 | 0 | 0 | 0.19 | 0.358 | 0 | 0.151 | 0 | 0.621 | 0 | 0 | 0.094 | 0.431 | 0 | 0.742 | 0 | 0 | 0.141 | 0.442 | 0.865 |
| 12 | 0.356 | 0 | 0 | 0 | 0.807 | 0 | 0.962 | 0 | 0.641 | 0 | 0 | 0 | 0 | 0 | 0.156 | 0.698 | 0 | 0.396 | 0 | 0.327 | 0 | 0 | 0.096 | 0.217 | 0 | 0.726 | 0 | 0 | 0.847 | 0.341 | 0.752 |
| 13 | 0.414 | 0 | 0 | 0 | 1 | 0 | 0.955 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0.237 | 0.459 | 0 | 0.199 | 0 | 0.697 | 0 | 0 | 0.031 | 0.669 | 0 | 0.371 | 0 | 0 | 0.553 | 0.28 | 0.668 |
| 14 | 0.435 | 0 | 0 | 0 | 0.819 | 0 | 0.878 | 0 | 0.324 | 0 | 0 | 0 | 0 | 0 | 0.201 | 0.825 | 0 | 0.549 | 0 | 0.165 | 0 | 0 | 0.106 | 0.439 | 0 | 0.52 | 0 | 0 | 0.686 | 0.129 | 0.592 |
| 15 | 0.479 | 0 | 0 | 0 | 0.961 | 0 | 1 | 0 | 0.686 | 0 | 0 | 0 | 0 | 0 | 0.208 | 0.733 | 0 | 0.364 | 0 | 0.277 | 0 | 0 | 0.08 | 0.394 | 0 | 0.693 | 0 | 0 | 0.334 | 0.115 | 0.528 |
| 16 | 0.49 | 0 | 0 | 0 | 0.5 | 0 | 0.981 | 0 | 0.57 | 0 | 0 | 0 | 0 | 0 | 0.149 | 0.457 | 0 | 0.63 | 0 | 0.12 | 0 | 0 | 0.03 | 0.216 | 0 | 0.666 | 0 | 0 | 0.881 | 0.329 | 0.438 |
| 17 | 0.509 | 0 | 0 | 0 | 0.818 | 0 | 0.973 | 0 | 0.579 | 0 | 0 | 0 | 0 | 0 | 0.365 | 0.25 | 0 | 0.392 | 0 | 0.588 | 0 | 0 | 0.059 | 0.368 | 0 | 0.636 | 0 | 0 | 0.721 | 0.222 | 0.497 |
| 18 | 0.575 | 0 | 0 | 0 | 0.838 | 0 | 1 | 0 | 0.722 | 0 | 0 | 0 | 0 | 0 | 0.96 | 0.328 | 0 | 0.432 | 0 | 0.345 | 0 | 0 | 0.028 | 0.184 | 0 | 0.924 | 0 | 0 | 0.709 | 0.206 | 0.395 |
| 19 | 0.645 | 0 | 0 | 0 | 0.87 | 0 | 0.94 | 0 | 0.986 | 0 | 0 | 0 | 0 | 0 | 0.134 | 0.099 | 0 | 0.478 | 0 | 0.388 | 0 | 0 | 0.098 | 0.299 | 0 | 0.43 | 0 | 0 | 0.878 | 0.242 | 0.234 |
| 20 | 0.626 | 0 | 0 | 0 | 0.835 | 0 | 0.966 | 0 | 0.676 | 0 | 0 | 0 | 0 | 0 | 0.307 | 0.118 | 0 | 0.508 | 0 | 0.325 | 0 | 0 | 0.18 | 0.506 | 0 | 0.444 | 0 | 0 | 0.525 | 0.074 | 0.245 |

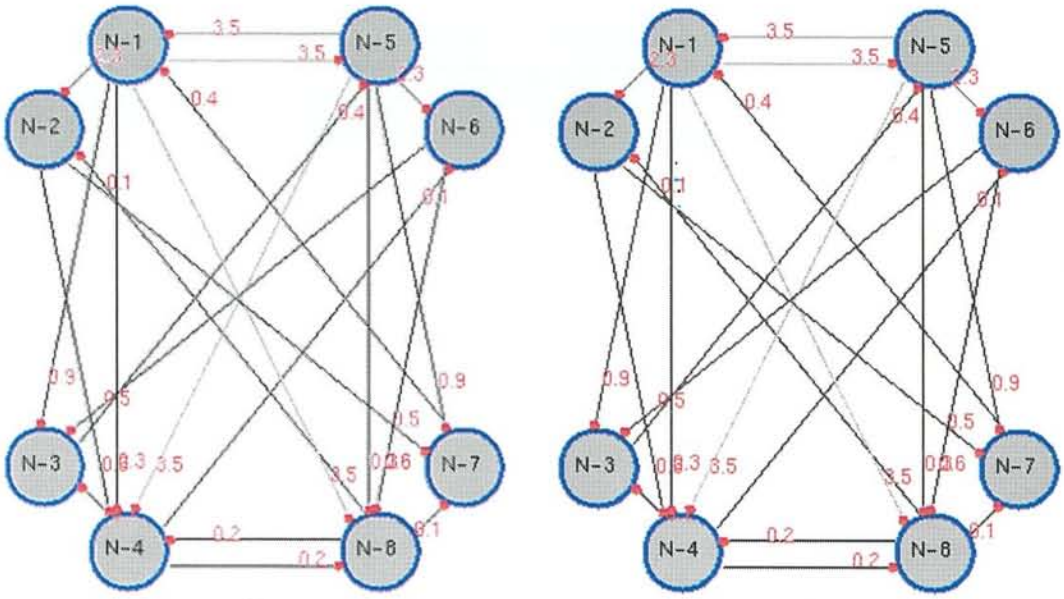



**Figure 5.42:** *Neuron connectivity structures a) 7th walking pattern; b) 14th walking pattern*

The samples of neuron connectivity structure are represented by the seventh and the fourteenth walking pattern depicted in Fig. 5.42 and those walking movements are illustrated in Fig. 5.43 where we can see, that the fourteenth walking pattern is faster than the seventh walking pattern.

#### 5.4.3.2 Gait Generator Learning

After all walking pattern parameters had been acquired, we used MLP to acquire the dynamical function of input (walking speed) and output of the walking pattern parameters resulted by walking patterns formation optimization. In this experiment, the activation function of neuron is sigmoid function as shown in Eq. (5.64), where $\alpha$ is defined as 4.0. We set variable $\eta$ in Eq. (5.63) as 0.1. In this experiment, it took $5 \times 10^6$ iterations until small error is acquired. We tried 3 models of MLP with different number of layers. First, second, and third MLP have 5 layers, 4 layers, and 3 layers, respectively with same number of weight calculations. In the first MLP, the number of neurons in the 1st–5th layers are 1, 7, 8, 30, 30, respectively; in the second MLP, the number of neurons in the 1st–4th layers are 1, 10, 30, 30, respectively; and in the third MLP, the number of neurons in the 1st–3rd layers are 1, 50, 30, respectively. The best number of neurons in each layer are determined by preliminary tests.

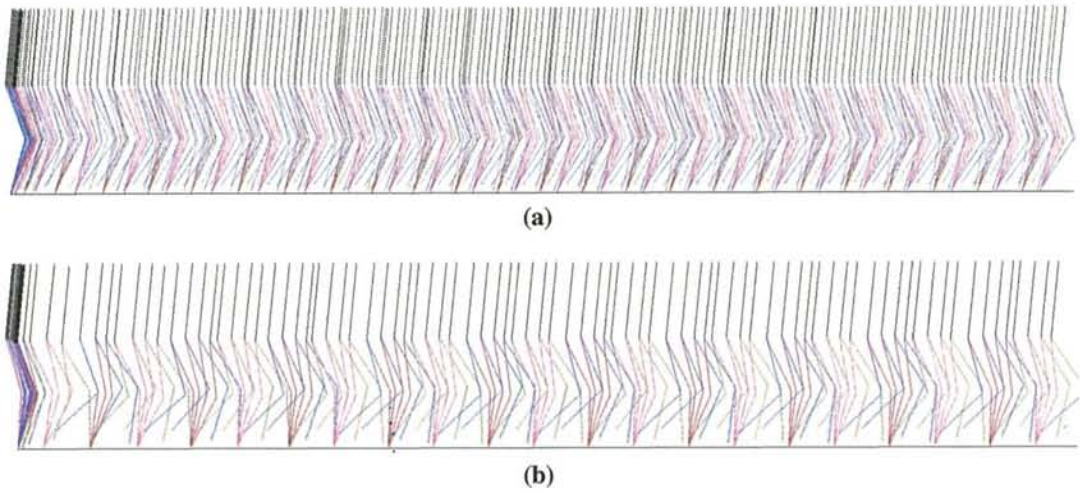


**Figure 5.43:** *Tracking of walking movements a) 7th walking pattern; b) 14th walking pattern*

$$f(x) = \frac{1}{1 + e^{-x/\alpha}} \tag{5.64}$$

The error values in this MLP's process are shown in Fig. 5.44. We used the optimized weight parameters of MLP as gait generator parameters for generating the walking parameters with desired walking speed. The MLP which has the larger number of layers has the better performance which can reach the smallest error. The best performing MLP is chosen as the gait generation structure model.

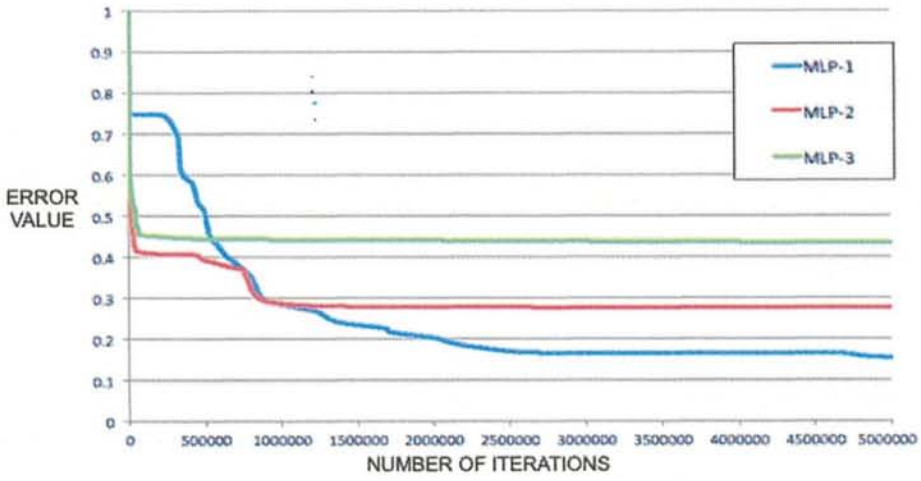



**Figure 5.44:** *Error evolution in different MLP structures*

#### 5.4.3.3 Walking Speed Control

In the first step, we implemented the system in Open Dynamics Engine computer simulation [183]. We put the weight values of the best MLP after optimization in the gait generator learning experiment and used that MLP model in this robot simulation. The robot simulation walk with several changes in the walking speed input is depicted in Fig. 5.45. By using the proposed method, we can control the walking speed of the robot based on biological approach.

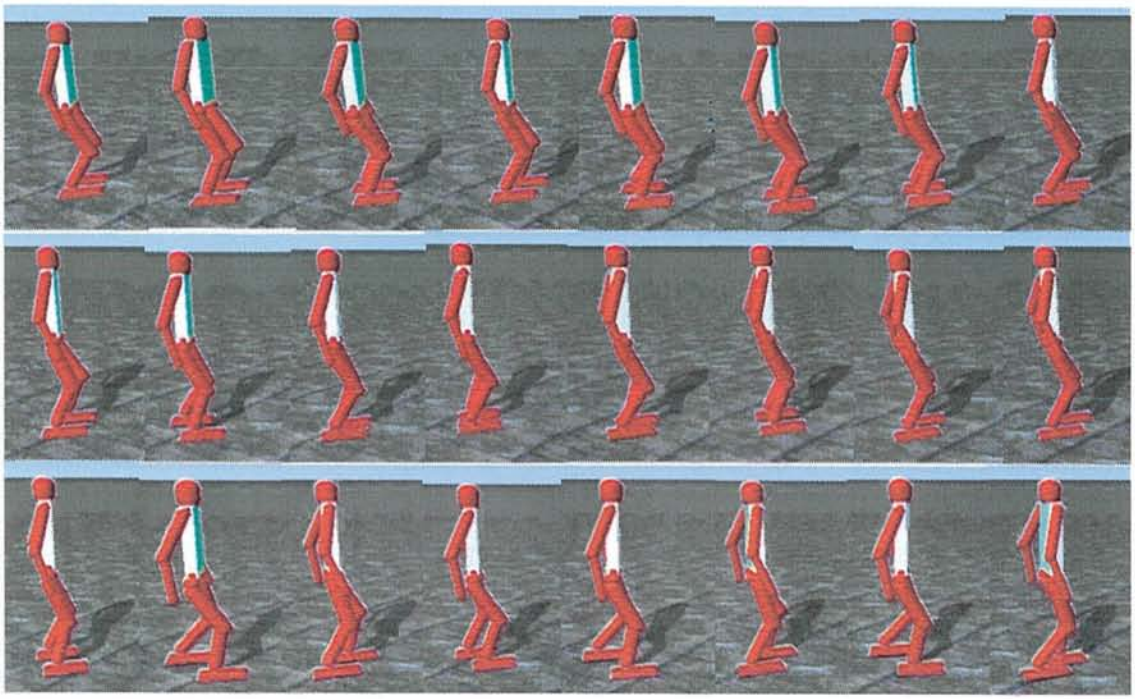


**Figure 5.45:** *Proposed locomotion implemented for humanoid robot in simulation level*

In this experiment, we took 2000 time sampling with different walking speeds in each 200 time sampling. The robot starts in silent condition with $d_v$ equals 0.0, and moves with different walking speed 0.31, 0.54, 0.74, 0.34, 0.10, 0.00, 0.32, 0.43, and 0.65, respectively. The signal change in this experiment is presented in Figs. 5.46 and 5.47. The neuron signal outputs changed smoothly depending on the change in walking speed. A change of speed of robot resulted from the walking experiment of robot is depicted in Fig. 5.48. The walking speed changed depending on the input of desired walking speed. Small oscillation in Fig. 5.48 are affected by the swinging motion of the legs when stepping. However, this condition does not affect the stabilization of robot walking.

In order to prove the high effectiveness of this walking speed control model, we implemented it in a simple humanoid biped robot shown in Fig. 7.18. We test the walking acceler-

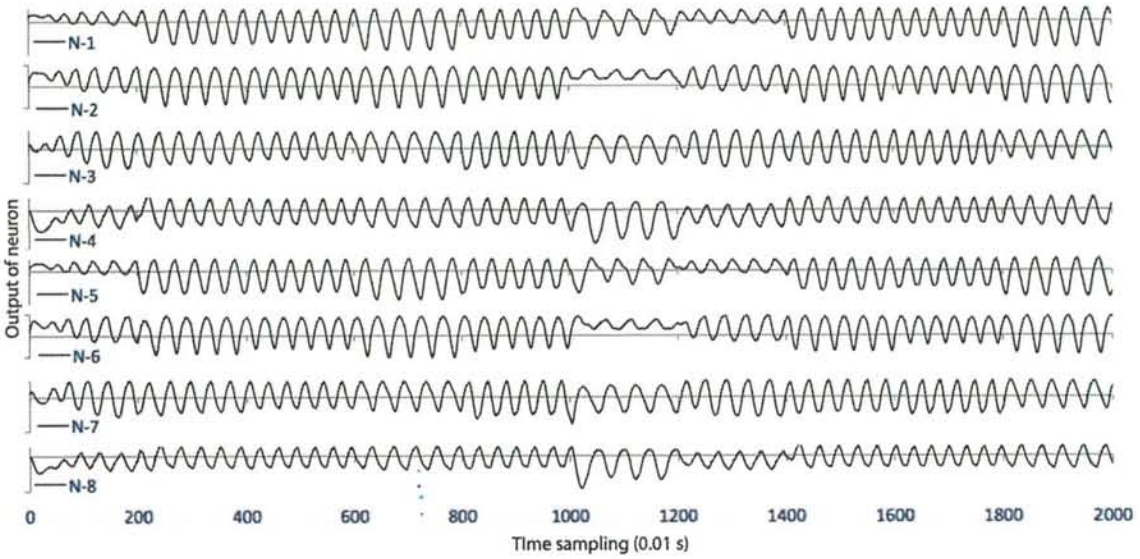



**Figure 5.46:** *The output of neuron response to the changing walking speed*

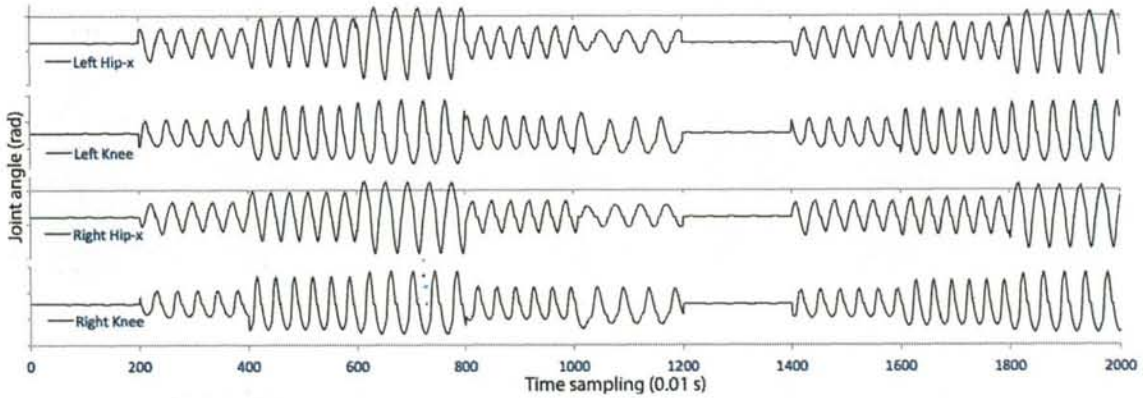



**Figure 5.47:** *The joint angle signal of the robot with the change in walking speed*

ation and deceleration performance. We set the walking speed of the robot with certain time. The biologically inspired robot locomotion is able to control the walking speed instead of the acceleration and deceleration control.

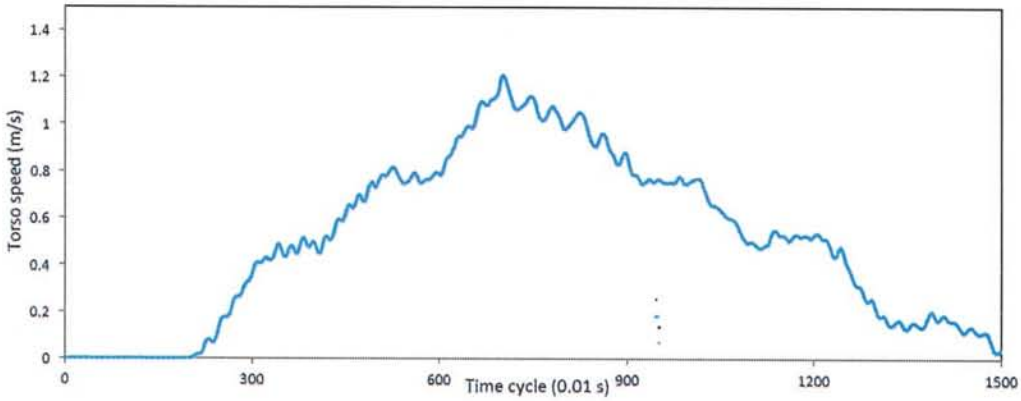



**Figure 5.48:** *A change of robot's torso walking speed*

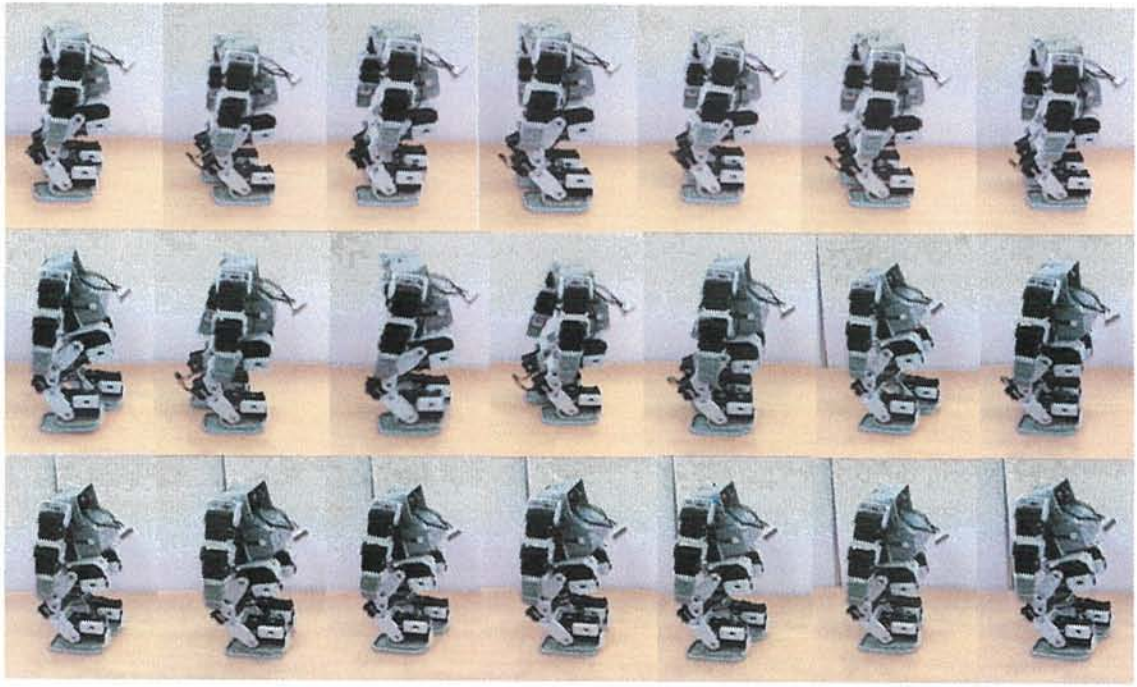


**Figure 5.49:** *Proposed locomotion implemented in a simple humanoid robot*

**5.4.3.3.1 Stability Test** In order to prove the stability of the proposed model, we analyzed the oscillation of the body tilt robot and angular velocity sensor placed at the torso of the robot. The oscillation of angular velocity recorded during walking speed control experiment

depicted in Fig.5.50 has stable oscillation. This represents, that the locomotion system is able to be applied to robot walking generation. Another substantiation that supports the proof of the robot stabilization is depicted in Fig.6.12, which presents the Poincare map analysis that shows the intersection of a periodic orbit in the state space of the angular velocity with the body tilt angle.

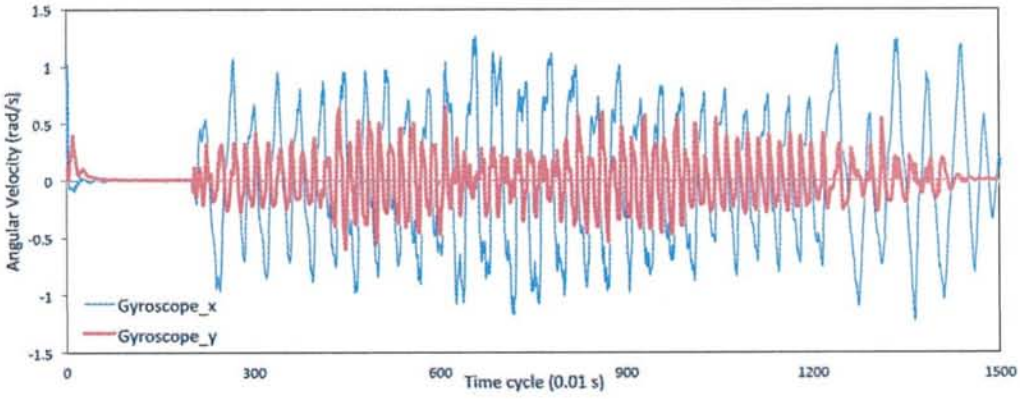



**Figure 5.50:** *Oscillation of angular velocity recorded from robot's body*

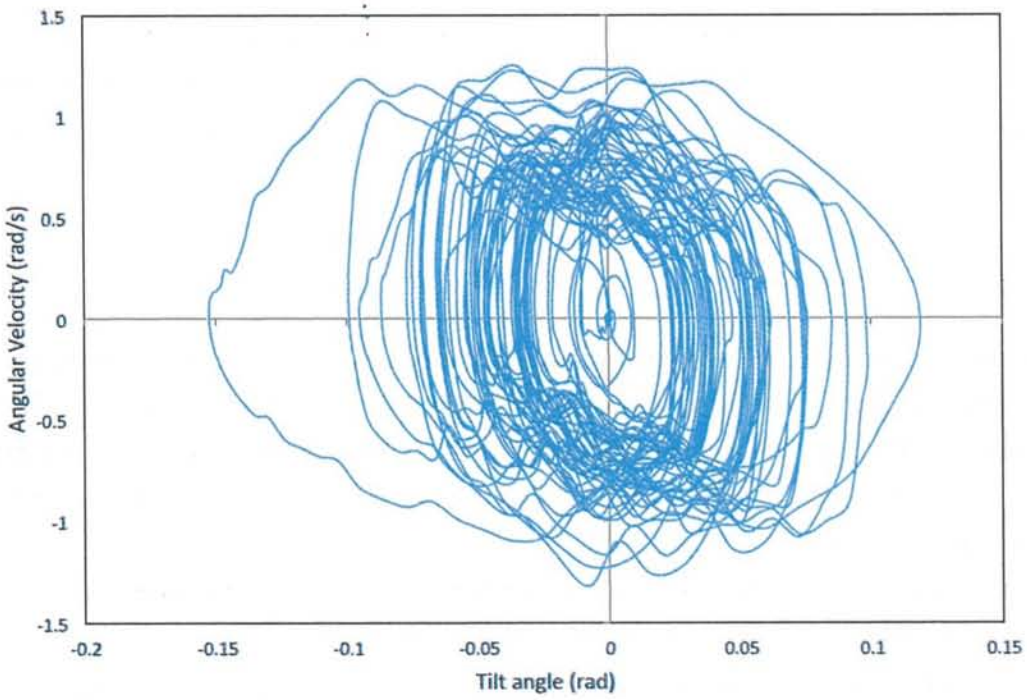



**Figure 5.51:** *Poincare diagram represents the stability of the robot*

### 5.4.4 Discussion

This research proposed a new walking speed control based on human behavior. We used neural oscillator as the locomotion generator and used evolutionary algorithm for forming several walking patterns with different walking speeds. The evolutionary algorithm with the proposed strategy effectively forms the walking pattern based on biological approach. In order to acquire the dynamical relationship between walking speed and its pattern, MLP is effectively implemented. The trained weight parameters in MLP are used for generating dynamical walking pattern by defining the desired walking speed in the gait generator system. By using the proposed locomotion model, the walking speed can be easily controlled. The locomotion generator can generate signal neuron transition smoothly with the change in walking speed. In order to prove the effectiveness of the proposed locomotion, we implemented it in open dynamics engine and in a simple humanoid robot.

This proposed locomotion model is the basic model of humanoid robot locomotion based on human behavior analysis. As a future work, the inverted pendulum will be combined with multimodal recurrent neural network for human behavior robust stability.

## 5.5 Omni-directional Locomotion Behavior

This section shows learning model of omni directional bio-inspired humanoid locomotion which is generating different walking behavior in different walking provision. Step length in sagittal and coronal direction, and degree of turning are considered parameters in walking provision.

### 5.5.1 Introduction

In this current issue of bio-inspired model, stability, walking provision (omni-directional walking), and learning process are obstacle in this locomotion development. Omni-directional walking are solved in this proposed research. Omni directional locomotion model provides dynamic movement and ability to modify its motion quickly so that support the robot to move in dynamic environment [12].

Most of the limit cycle development only consider speed in unidirectional walking [77, 52, 68]. Endo *et al* developed adjusted walking velocity for neural oscillator based locomotion. However, range of adjusted velocity is small [52]. In 2008, Manoonpong *et al* developed neural based locomotion in order to generate omnidirectional movement in any types of legged robot. The proposed model can easily be adapted to control other kinds of

walking machine without changing the internal network structure and its parameters. This model also can produce at least 11 different walking patterns and a self-protective reflex by using five input neurons [124]. Therefore in this research, we proposed dynamic structure model in neural oscillator based locomotion and learning model in order to generate unscaled omnidirectional movement in biped robot.

Proposed omni-directional walking cover combined 180 degree in horizontal walking direction and turning walking level. In our previous development, straight walking control and turning walking level have been developed independently in [171], [168], and [170], respectively. Behavior forming this various walking provision, will be learned by using evolutionary algorithm.

In this research, natural locomotion processes are applied, where neural based locomotion is designed with considering 4 degree of freedom in each legs. One joint is represented by 2 coupled neuron which imply the flexor and extensor muscle mechanism in human joint. Locomotion model is able to generate dynamic walking behavior in omni-directional walking provision. The main contribution of our proposed research are 1) Development of motor neuron interconnection structure in neural based locomotion for generating omni-directional walking controller 2) The novelty in learning methodology which proposed centered learning strategy for avoiding the complex interconnection neuron structure.

### 5.5.2 Centered Learning Model

The aim of this learning model is for learning the walking behavior of biped robot in unscaled omni-directional movement and avoiding the complexity of neural interconnection. Omni-directional locomotion based on neural oscillator in this proposed research is separated by 2 interconnections depicted in Fig. 6.24, where main interconnection structure represents sagital walking behavior which has important role for generating the walking phase and pattern, and branch interconnection structure represents turning and coronal behavior.

In centered learning model for omni-directional movement, we used modified multi layer perceptron (MLP) which given some initial relationship walking pattern references generated from behavior learning system in Section 5.5.4. The mathematical equations of the proposed MLP has been explain in [171]. In order to optimize every walking pattern reference, we used multi objective evolutionary algorithm. Online terminology representing online reference data will be updated from the walking parameter generated by optimized learning model and walking analysis parameter. Reference data in this learning model, compose 11 input walking provision and 72 output parameter representing walking parameter. Since the reference data will be updated every walking generation, the error resulted between desire

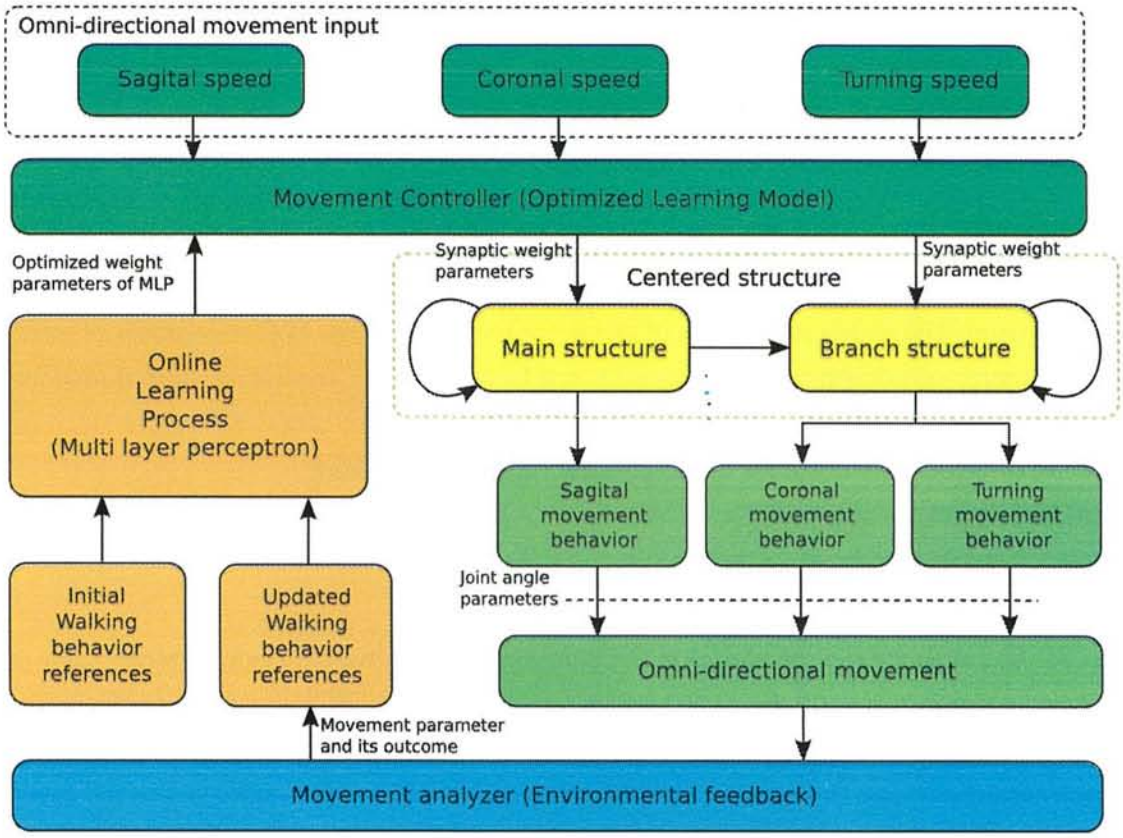



**Figure 5.52:** *Online model of centered interconnection structure of omni-directional neuro based locomotion model*

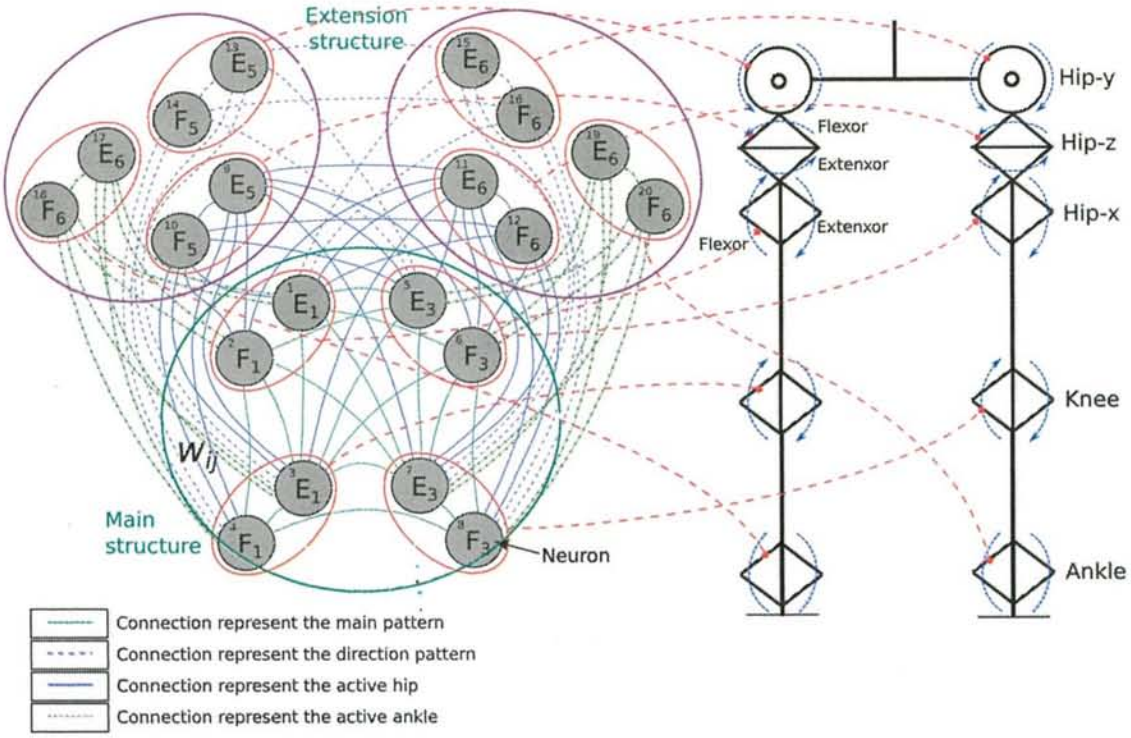



**Figure 5.53:** *Whole structure of centered based interconnection model of neural based locomotion*

walking provision and the result of walking which its parameters generated learning model will be reduced. In this model, unscaled omni-directional control and combination of walking behaviors are learned. Therefore after learned, walking provision in omni-directional movement without scalling parameter can be as the input parameter in locomotion generator. The synaptic weights in neural structure are generated based on the input of walking provision.

Furthermore, locomotion generator based on neural oscillator in angle joint level is conducted. The neuro-based locomotion generator will be explained in Section 5.5.3.

### 5.5.3 Neuro-based Locomotion Generator

Our bio-inspired locomotion uses neural oscillator model proposed by Matsuoka [130]. This model is generated by mutual inhibition between certain neurons. Each neurons also acquires adaptation signal input. In our previous research, there were 12 motor neurons to represent 6 joints [175, 171]. In order to realize the omni-directional movement in biped robot, additional joints are required. This proposed model increases the number of neurons to 16 neurons, representing 4 joints in each legs which can be seen in Fig. 5.53. The mathematical model of the proposed neuro-locomotion system is shown in Eqs. (5.65), (5.66),

(5.67).

$$\tau \dot{u}_i = (u_0 - u_i - \sum_{j=1}^{n} w_{ij} y_j - b v_i)\tau_f \tag{5.65}$$

$$\tau' \dot{v}_i = (y_i - v_i)\tau_f \tag{5.66}$$

$$y_i = \max(u_i, 0) \tag{5.67}$$

$$u_i' = u_i g_i \tag{5.68}$$

where $u_i$, $y_i$, and $v_i$ are the inner state, output value, and a variable that represent the adaptation value or the self-inhibition effect of the $i$th neuron, respectively; $b$ is the rate of the adaptation value. The external input for coupled neurons, which have a constant rate, is denoted by $u_0$. The time constant of the inner state and the adaptation effect in the neuron are represented by $\tau$ and $\tau'$, respectively. In Eq. (5.65), $w_{ij}$ represents the strength of the inhibitory effect between the motor neurons that is optimized offline; $\sum_{j=1}^{n} w_{ij} y_j$ represents the total of the signal input from the neuron. In Eqs. (5.65) and (5.66), $\tau_f$ is used for controlling the frequency of oscillation.

The synaptic weight $(w_{ij})$ will be optimized depending on the desired walking behavior. Since there are 16 motor neurons involved in the locomotion generator, there are $16^2$ synaptic weight combination from $(w_{1,1} - w_{16,16})$, where its interconnection map from source neuron to destination neuron can be seen in Table 5.14. There are ... parameters $(G_1, G_2, G_3, \cdots, G_{...})$ representing the synaptic weight to be optimized. For example, parameter $G_{24}$ represents the weight connection from neuron 7 to neuron 8 and from neuron 4 to neuron 3. twenty eight parameters $(G_1, G_2, G_3, \cdots, G_{28})$ represent the walking behavior in sagittal direction, 23 parameters $(G_29, G_30, \cdots, G_{51})$ represent the behavior in coronal direction, and 23 parameters $(G_52, G_53, \cdots, G_{74})$ represent the turning behavior. The neuron structure representing walking behavior in sagittal direction is constructed as the main structure of interconnection structure. This neuron will effect to neurons representing coronal behavior and turning behavior, but neurons representing coronal behavior and turning behavior are not affected each others.

### 5.5.4 Behaviors Learning System

The aim of this system is to generate the behaviors for representing omni-directional movement. In this section, walking behavior and optimization strategy are explained. In order to build omni-directional movement, at least there are 3 parameters, sagittal movement, coronal movement, and turning movement [187, 184]. Therefore, in the behavior building

**Table 5.14:** *Interconnection Maps Represented by Genes*

| | | Destination Neuron | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 |
| Source Neuron | 1 | 0 | $G_1$ | $G_2$ | $G_3$ | $G_4$ | $G_5$ | $G_6$ | $G_7$ | $G_{29}$ | $G_{37}$ | $G_{33}$ | $G_{41}$ | $G_{51}$ | $G_{59}$ | $G_{55}$ | $G_{63}$ |
| | 2 | $G_8$ | 0 | $G_9$ | $G_{10}$ | $G_{11}$ | $G_{12}$ | $G_{13}$ | $G_{14}$ | $G_{30}$ | $G_{38}$ | $G_{34}$ | $G_{42}$ | $G_{52}$ | $G_{60}$ | $G_{56}$ | $G_{64}$ |
| | 3 | $G_{15}$ | $G_{16}$ | 0 | $G_{17}$ | $G_{18}$ | $G_{19}$ | $G_{20}$ | $G_{21}$ | $G_{31}$ | $G_{39}$ | $G_{35}$ | $G_{43}$ | $G_{53}$ | $G_{61}$ | $G_{57}$ | $G_{65}$ |
| | 4 | $G_{22}$ | $G_{23}$ | $G_{24}$ | 0 | $G_{25}$ | $G_{26}$ | $G_{27}$ | $G_{28}$ | $G_{32}$ | $G_{40}$ | $G_{36}$ | $G_{44}$ | $G_{54}$ | $G_{62}$ | $G_{58}$ | $G_{66}$ |
| | 5 | $G_{25}$ | $G_{26}$ | $G_{27}$ | $G_{28}$ | 0 | $G_{22}$ | $G_{23}$ | $G_{24}$ | $G_{33}$ | $G_{41}$ | $G_{29}$ | $G_{37}$ | $G_{55}$ | $G_{63}$ | $G_{51}$ | $G_{59}$ |
| | 6 | $G_{17}$ | $G_{18}$ | $G_{19}$ | $G_{20}$ | $G_{21}$ | 0 | $G_{15}$ | $G_{16}$ | $G_{34}$ | $G_{42}$ | $G_{30}$ | $G_{38}$ | $G_{56}$ | $G_{64}$ | $G_{52}$ | $G_{60}$ |
| | 7 | $G_9$ | $G_{10}$ | $G_{11}$ | $G_{12}$ | $G_{13}$ | $G_{14}$ | 0 | $G_8$ | $G_{35}$ | $G_{43}$ | $G_{31}$ | $G_{39}$ | $G_{57}$ | $G_{65}$ | $G_{53}$ | $G_{61}$ |
| | 8 | $G_1$ | $G_2$ | $G_3$ | $G_4$ | $G_5$ | $G_6$ | $G_7$ | 0 | $G_{36}$ | $G_{44}$ | $G_{32}$ | $G_{40}$ | $G_{58}$ | $G_{66}$ | $G_{54}$ | $G_{62}$ |
| | 9 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | $G_{48}$ | $G_{49}$ | $G_{45}$ | 0 | 0 | 0 | 0 |
| | 10 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | $G_{45}$ | 0 | $G_{50}$ | $G_{46}$ | 0 | 0 | 0 | 0 |
| | 11 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | $G_{46}$ | $G_{49}$ | 0 | $G_{47}$ | 0 | 0 | 0 | 0 |
| | 12 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | $G_{47}$ | $G_{50}$ | $G_{48}$ | 0 | 0 | 0 | 0 | 0 |
| | 13 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | $G_{70}$ | $G_{71}$ | $G_{67}$ |
| | 14 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | $G_{67}$ | 0 | $G_{72}$ | $G_{68}$ |
| | 15 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | $G_{68}$ | $G_{71}$ | 0 | $G_{69}$ |
| | 16 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | $G_{69}$ | $G_{72}$ | $G_{70}$ | 0 |

processes, behavior of 3 walking parameters are optimized in different value of each parameters.

#### 5.5.4.1 Behavior Building

Since the neuro based locomotion are implemented, behavior building implies to build the neuron interconnection. Different structure will generate different walking behavior. In this part, we will explain the strategy for optimizing the interconnection structure depending on the desired walking behavior. We propose multi-objective evolutionary algorithm NSGA-II, which firstly proposed by Dep *et al* [185, 36] which was proved has optimized neuro-based locomotion [178]. In this behavior forming, the optimization processes are separated by 3 parts, forming sagittal direction behaviors walking in different speed, forming coronal direction behaviors in different speed, and turning behaviors in different degree of turning level.

**5.5.4.1.1 Sagittal direction behaviors optimization** This interconnection optimization is the improvement from our previous researches [175, 171] in optimization part. Optimization objective is improved from single objective to multi-objective optimization and also the encoding process. We expect to build the walking behavior with stable pelvis speed with desired length of step and height of step.

At the first stage, we optimized the normalize of initial walking behavior ($x^{(0)}_{1-28}$) by using evolutionary algorithm in [175]. Then, we optimize the walking behavior in each speed unit based on initial behavior. In this NSGA optimization, one chromosome has 28 genes ($x_{1-28}$) converted to walking pattern ($g_{1-28}$) which are representing the synaptic weight value and the neuron structure. The genes have minimum and maximum value, $x^{(a)}_{min}$ and $x^{(a)}_{max}$, which are decided depending on preliminary test. The conversion equation from the genes to the interconnection parameter is shown in Eqs. (5.69) and (5.70), where $x_j$ is the $j$th gene of the solution in normalized state and $G_j$ is the $j$th gene that is ready to be used in the neural locomotion. If $g_j$ is lower than zero, then, $g_j$ for synaptic weight is zero implying that there is no connection between the neurons. $g_j$ parameter will be converted to synaptic weight parameter based on map connection depicted in Table 5.14.

$$g_j = \begin{cases} g'_j & \text{if } x_j > 0 \\ 0 & \text{otherwise} \end{cases} \tag{5.69}$$

$$g'_j = x_j(x^{(a)}_{max} - x^{(a)}_{min}) + x^{(a)}_{min} \tag{5.70}$$

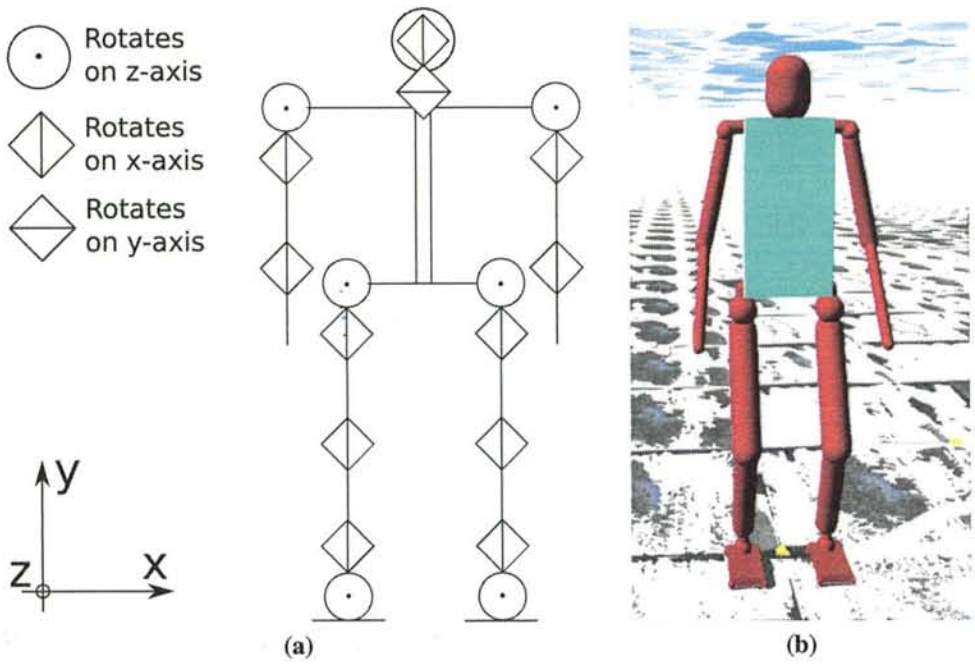



**Figure 5.54:** *(a) Robot design from front side (b) Robot design from side.*

In the fitness model, we minimize the error of pelvis speed, error of step length, and error of step height. Error of pelvis speed $E_v^{(a)}$ is calculated as $E_v^{(a)} = \sum_{t=1}^{T}(d_x^{(v)} - v_x(t))^2$, where $t$ and $T$ are the time unit and maximum time, $d_x^{(v)}$ is the desired speed, and $v(t)$ is the current speed in sagittal direction. Error of step length $(E_l^{(a)})$ is calculated as $E_l^{(a)} = \sum_{s=1}^{N_s}(d^{(l)} - l(s))^2$, where, parameter $s$ and $N_s$ are the current step and maximum step resulted until maximum time $(T)$, $d^{(l)}$ is the desired length of step, and $l(t)$ is the length of step in $s$th step. Error of step height $(E_h^{(a)})$ is calculated as $E_h^{(a)} = \sum_{s=1}^{N_s}(d^{(h)} - h(t))^2$, where, parameter $d^{(h)}$ is the desired height of step and $h(s)$ is the height of step in $s$th step. Considered parameters are grouped into 2 fitness functions are calculated as minimization shown in Eqs. (5.71) and (5.72).

$$g_{1-28} = \arg\min_{w_{i,j}}(E_v^{(a)} + E_l^{(a)}) \tag{5.71}$$

$$g_{1-28} = \arg\min_{w_{i,j}} E_h^{(a)} \tag{5.72}$$

This optimization will be conducted $P+1$ times with different walking speed, where $d_0^{(v)}$ is minimum desired speed $v_min$ and $d_{P+1}^{(v)}$ is maximum desired speed $v_{max}$ where the speed iteration is $(v_{max} - v_{min})/P$.

**5.5.4.1.2 Coronal direction behaviors optimization** In this interconnection optimization, we expect to build the walking behavior with desired step length in coronal direction. Since this optimization used the optimized weight parameter of sagittal direction ($g_{1-28}$) in each level of speed, this optimization will be also conducted $P+1$ times. One chromosome has 22 genes ($x_{29-50}$) converted to walking pattern ($g_{29-50}$) which are representing the synaptic weight value and the neuron structure. The genes have minimum and maximum value, $x_{min}^{(b)}$ and $x_{max}^{(b)}$, which are decided depending on preliminary test. The conversion equation from the genes to the interconnection parameter is shown in Eqs. (5.69), where $g_j' = x_j(x_{max}^{(b)} - x_{min}^{(b)}) + x_{min}^{(b)}$.

In the evaluation model, we minimize the error of pelvis speed in coronal direction, error in length of step in coronal direction, and error of center of mass. The error of pelvis speed in coronal direction $E_{v_y}^{(b)}$ is calculated as $E_{v_y}^{(b)} = \sum_{t=1}^{T}(d_y^{(v)} - v_y(t))^2$, where, $t$ and $T$ are the time unit and maximum time, $d_y^{(v)}$ is the desired speed, and $v_y(t)$ is the current speed in coronal direction. The error in length of step in coronal direction $E_{l_y}^{(b)}$ is calculated as $E_{l_y}^{(b)} = \sum_{s=1}^{N_s}(d^{(l)_y} - l_y(s))^2$, where, $d^{(l)_y}$ is the desired length of step, and $l_y(s)$ is the length of step in $s$th step in coronal direction. and error of center of mass $E_{CoM}^{(b)}$ is calculated as $E_{CoM}^{(b)} = \sum_{t=1}^{T}(y_{CoM}(t) - y_{pelvis}(t))^2$ where, parameter $y_{CoM}(t)$ is the center of mass and $y_{pelvis}$ is the pelvis position in coronal direction. Considered parameters are grouped into 2 fitness functions are calculated as minimization shown in Eqs. (5.73) and (5.74).

$$g_{29-51} = \arg\min_{w_{i,j}}(E_{v_y}^{(b)} + E_{l_y}^{(b)}) \tag{5.73}$$

$$g_{29-51} = \arg\min_{w_{i,j}} E_{CoM}^{(b)} \tag{5.74}$$

**5.5.4.1.3 Turning direction behaviors optimization** Neuron representing the turning behavior will be optimized in each speed level. This optimization will be also conducted $P+1$ times depending on the changing of neurons structure representing sagittal direction ($g_{1-28}$). Since these neurons are not affected by neuron representing coronal direction, parameter $g_{29-50}$ are not used in this optimization. One chromosome has 22 genes ($x_{51-72}$) converted to walking pattern ($g_{51-72}$) which are representing the synaptic weight value and the neuron structure. The genes have minimum and maximum value, $x_{min}^{(c)}$ and $x_{max}^{(c)}$, which are decided depending on preliminary test. The conversion equation from the genes to the interconnection parameter is shown in Eqs. (5.69), where $g_j' = x_j(x_{max}^{(c)} - x_{min}^{(c)}) + x_{min}^{(c)}$.

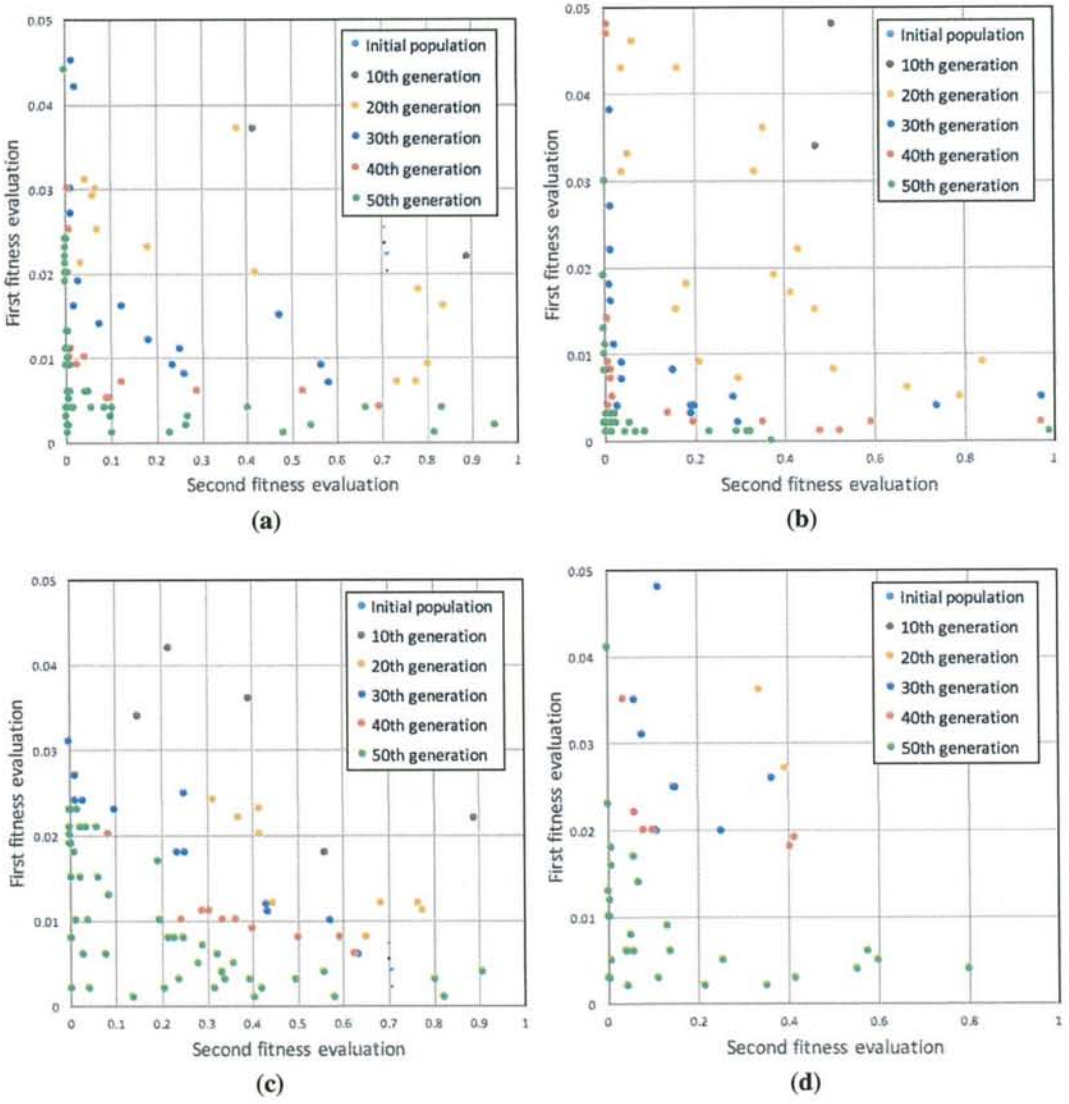



**Figure 5.55:** *Sample of pareto front in different generation (a) 3 m/s of desire walking speed (b) 5 m/s of desire walking speed (c) 7 m/s of desire walking speed (d) 9 m/s of desire walking speed.*

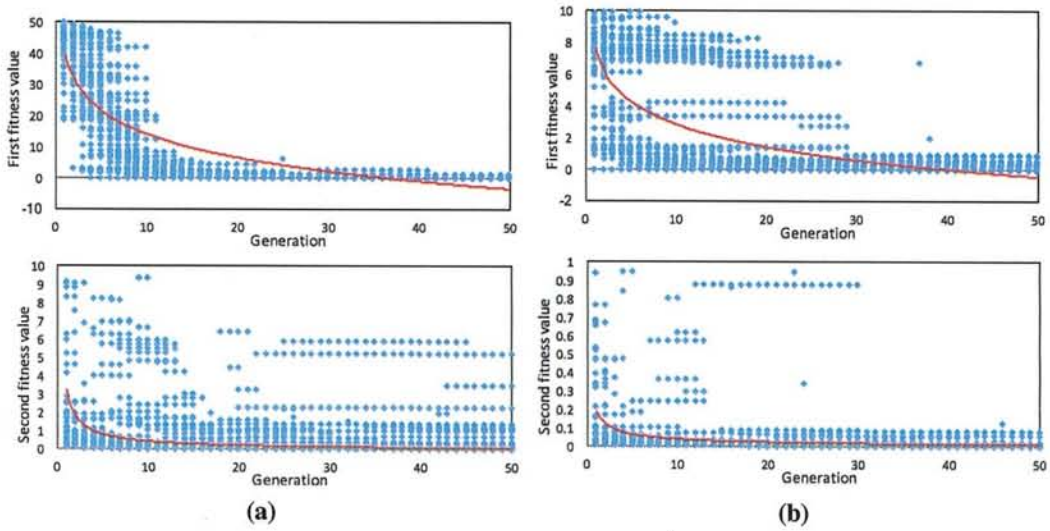



**Figure 5.56:** *Sample of fitness evolution of population in every generation (a) with 5 m/s of desire walking speed (b) with 7 m/s of desire walking speed*

$$E_t^{(c)} = \begin{cases} \alpha - d_\alpha & \text{if } s\%2 = 0 \\ 0 & \text{otherwise} \end{cases} \tag{5.75}$$

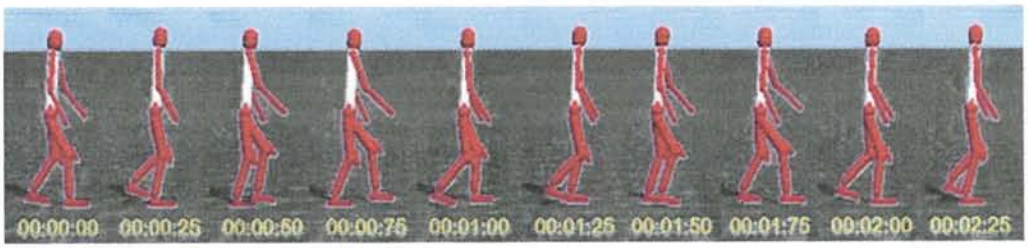
$$E_a^{(c)} = \sum_{t=1}^{T} (\alpha(t) - \alpha(t-1))^2 \tag{5.76}$$

In the evaluation process, we minimize the error of the walking direction $E_t^{(c)}$ which is calculated in Eqs. (5.75), (5.76). In Eq. (5.75), $d_\alpha$ is the desired turning angle and $\alpha = \theta_(l) - \theta_(r)$, where, $\theta_(l)$ and $\theta_(r)$ are the angle value of hip-z joint in left and right leg, respectively. Second evaluation, we consider to minimize turning speed which is calculated in Eq. (5.76).

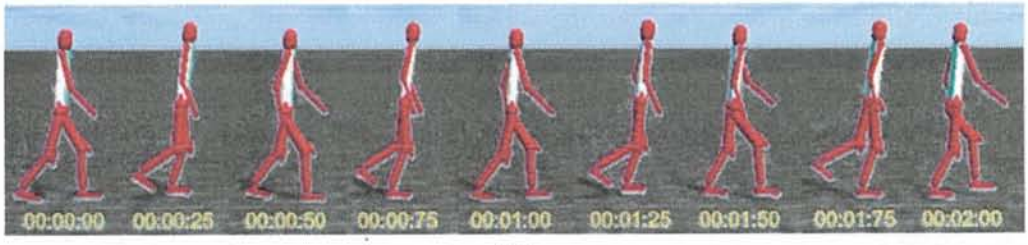
$$g_{52-74} = \arg\min_{w_{i,j}} \left( \sum_{s=1}^{N_s} E_t^{(c)}(s) \right) \tag{5.77}$$

$$g_{52-74} = \arg\min_{w_{i,j}} \left( E_a^{(c)} \right) \tag{5.78}$$

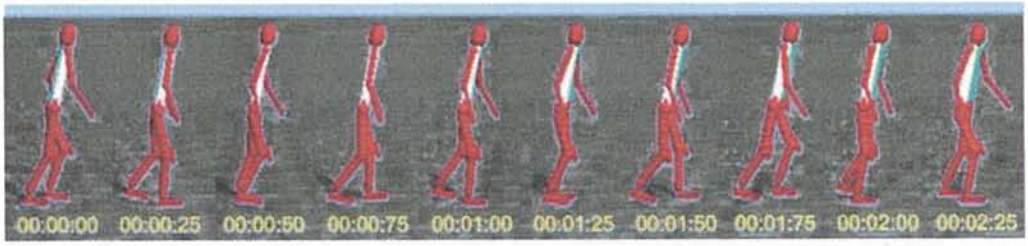
Fitness evaluations in this optimization process are calculated as minimization shown in Eqs. (5.77) and (5.78).



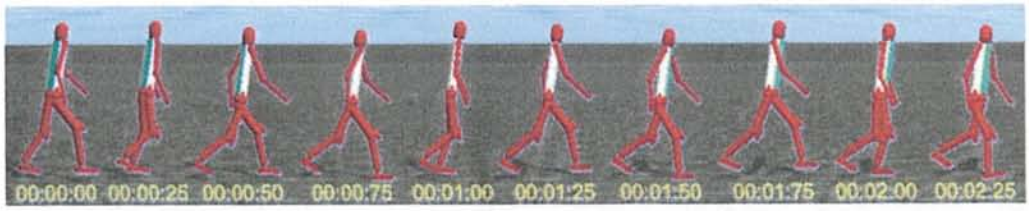

(a)




(b)




(c)




(d)

**Figure 5.57:** *Generated sagittal movement behavior in different speed (a) normal walking behavior in low speed (b) fast walking behavior in medium speed (c) jogging behavior in medium speed (d) running behavior in high speed movement*

### 5.5.5 Experimental Result and Discussion

In order to prove the defectiveness of the proposed model, we implemented the proposed model into computer simulation and using Open Dynamics Engine as the physics engine. We designed biped robot with 4 DoF in each legs which can be seen in Fig. 6.14. In first experiment, we optimized the interconnection structure respected to sagittal direction as the center structure in order to get different speed of walking behavior. In second experiment, the branch structures (Coronal direction and turning behavior) and their interconnection with center structure will be optimized.

#### 5.5.5.1 Center Interconnection Structure Optimization

In this optimization, we generated the walking pattern in different walking speed between 0m/s - 5m/s with iteration 0.2 m/s unit speed. We will generate the walking pattern with minimum energy required in every desire walking speed. The parameter of NSGA-II used in this optimization process can be seen in Table. 5.15. We had 64 individuals in one population, where one individual composed 28 gene parametes representing synaptic weight in main interconnection. We run the evolution process until 50 generations and took the best individudl in pareto front in every desired walking speed as the walking pattern references. The sample of evolutionary process can be seen in Figs. 5.55 and 5.56 which show the increasing of pareto front's diversity. They also show that the population is approaching the minimum error value in both fitness evaluation. Therefore, the optimization will generate the desired movement with minimum energy required.

**Table 5.15:** *NSGA-II Parameters*

| Parameter | Value |
|---|---|
| Population size | 64 indv. |
| Number of generations | 50 generations |
| Number of objectives | 2 objectives |
| Chromosome | 28 real numbers |
| Crossover & Mutation prob. | 0.5, 0.5 |
| Time evaluation | 200 time sampling, @20ms |

From this experiment, we took the best pareto front solution in every desired sagital speed. Therefore, one desired speed may generate more than one solution with different structure. Since we had 26 desired sagital speed, we acquired 26 unique interconnection patterns. The sample of sagital behavior can be seen in Fig. 5.57.

#### 5.5.5.2 Branch Interconnection Structure Optimization

Since the main interconnection structure has been optimized, we continued to optimize the brance structure. There are 2 movement behaviors to be optimized, coronal and turning movement behaviors. We used optimized main interconnection structure in every desired speed for optimizing branch interconnection structure.

**5.5.5.2.1 Coronal Movement Behavior** In coronal movement behavior, we generated 9 desired coronal movement speed in every interconnection patterns, from 0.8 m/s to -0.8 m/s with interval 0.2 m/s. We chose the best solution in pareto front from with minimum desire speed error in every optimization. Therefore, this coronal optimization we generated 234 interconnection structures. The sample of turning behavior can be seen in Fig. 5.58.

**5.5.5.2.2 Turning Movement Behavior** In turning movement behavior, we generated 9 desired turning movement speed in every interconnection patterns, from $-90^0/s$ until $90^0/s$ with turning speed interval $22.5^0/s$. We also chose the best solution in pareto from with minimum desire speed error in every optimization. Therefore, optimization processes also generated 234 interconnection structures. The sample of turning behavior can be seen in Fig. 5.59.

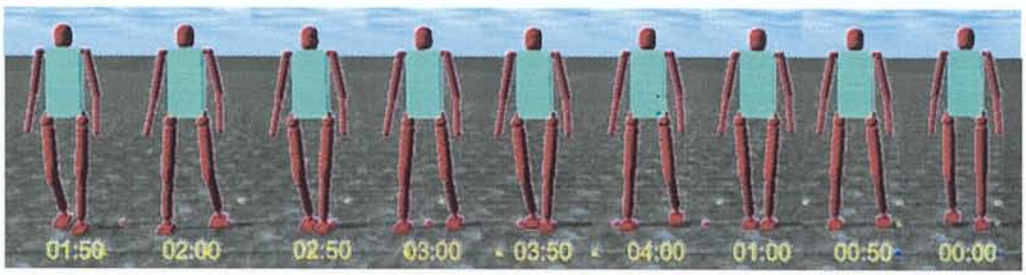
Since, every optimization behavior in branch structures had 234 interconnection structures, we got 468 behavior references in different movement provision.

#### 5.5.5.3 Learning of Omni-directional Walking Behavior

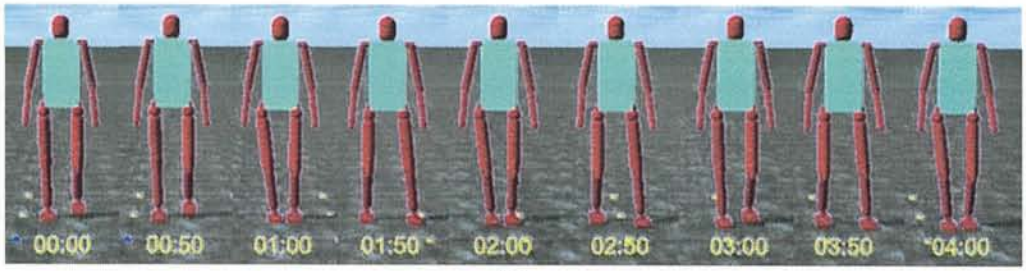
After having generated behavior references in every movement behavior, we conducted the unscaled omni-directional motion behavior by using proposed MLP. In this learning process, we would like to achieve the relationship between walking input (degree of turning, sagittal speed, coronal speed, phase different of 8 joint angle signals) and the interconnection structure of neural oscillator. There were 11 neurons in input layer, 20 neurons in every hidden layer, and 72 output layer.

In order to decrease the complexity, we divided the MLP system into 36 system. Therefore, every MLP had 2 neurons in output layer, where the structure can be seen in Fig. 5.60. In order to generalize the system, we used k-fold cross validation, where the $k$ value is 5. We trained the training data (walking references data) until $2.10^6$ iterations and the relationship error can be decreased as shown in Fig.5.62.

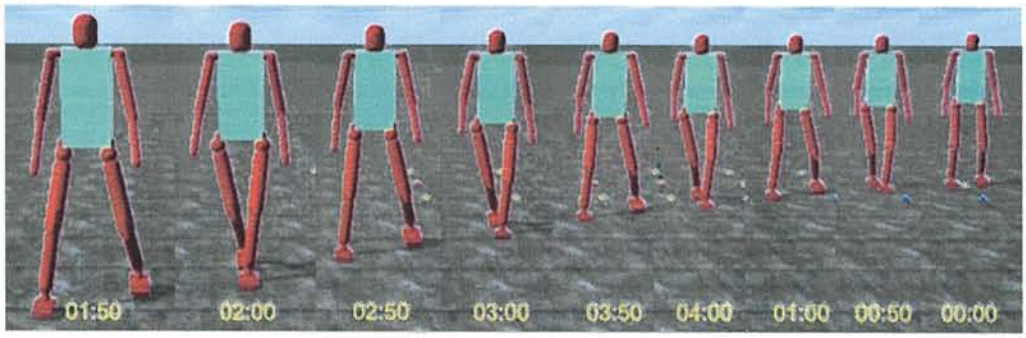
Using this proposed learning model, dynamic signal in unscaled omni-directional walking controller can be achieved. Fig.5.61a show the neuron signal outputs with movement



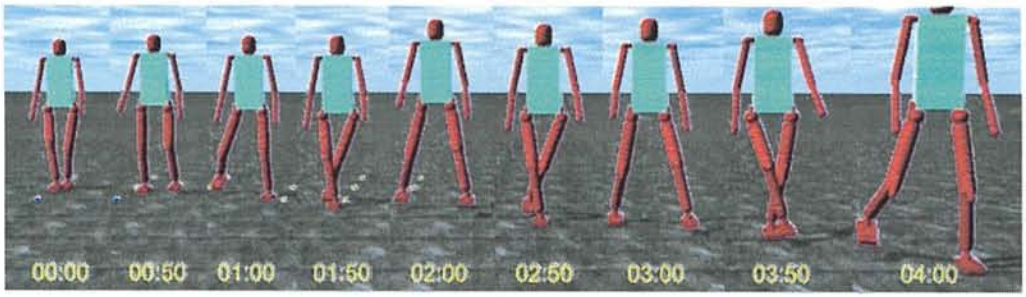

(a)




(b)




(c)




(d)

**Figure 5.58:** *Generated coronal movement behavior in different speed (a) slow coronal walking behavior in right direction (b) left direction (c) fast coronal walking in right direction (d) left direction*

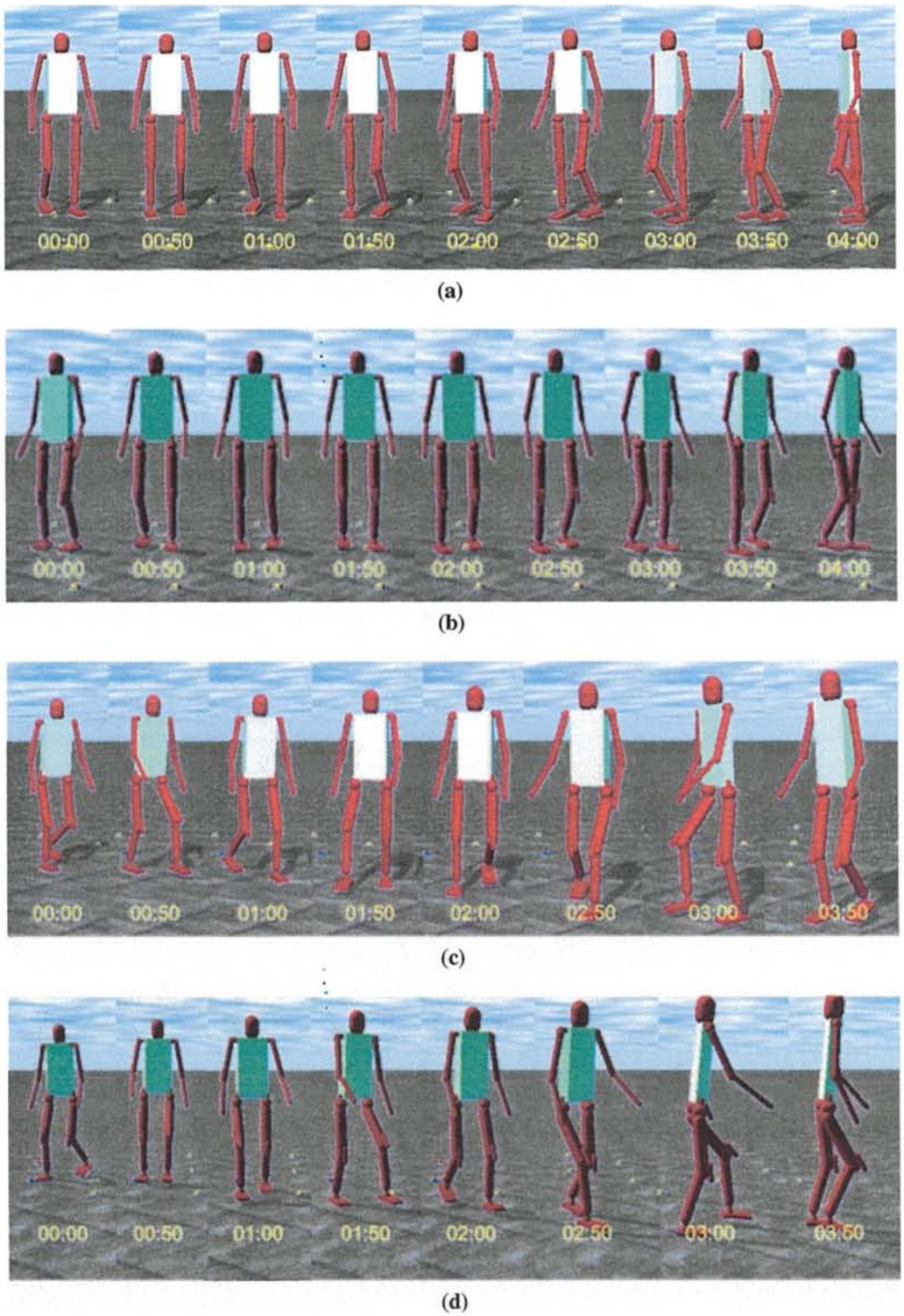



**Figure 5.59:** *Generated turning movement in different behavior (a) turning right on a place (b) turning left on a place (c) walking with turning right (d) walking with turning left*

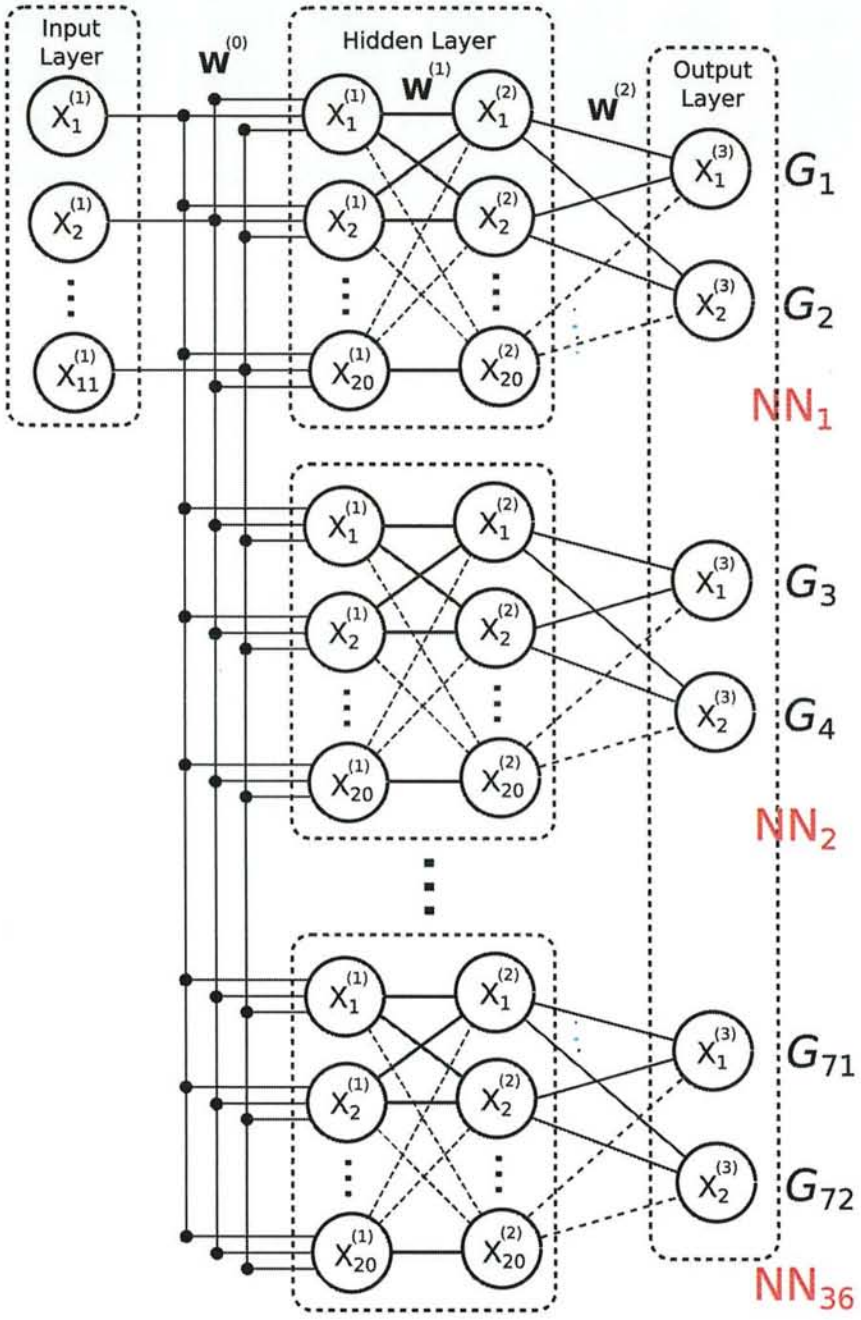
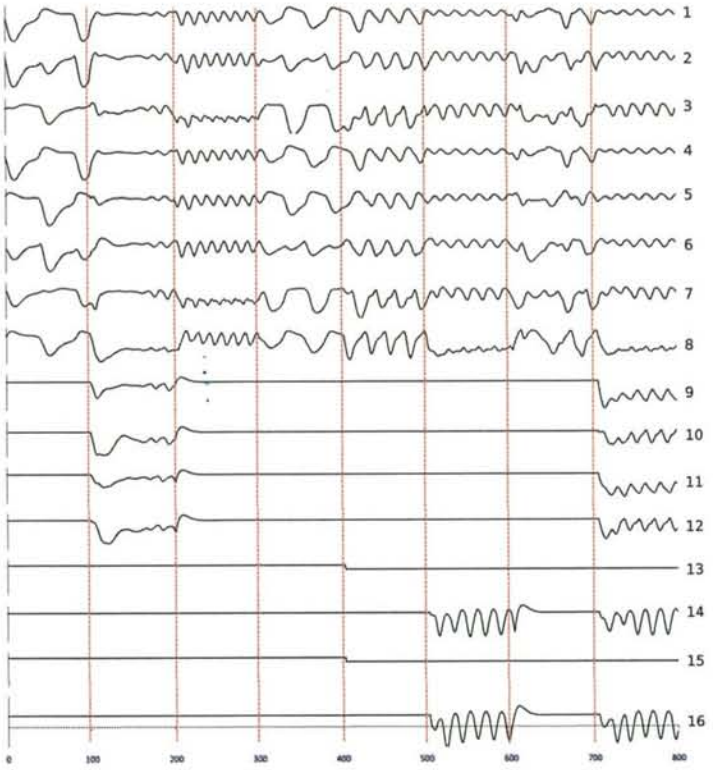
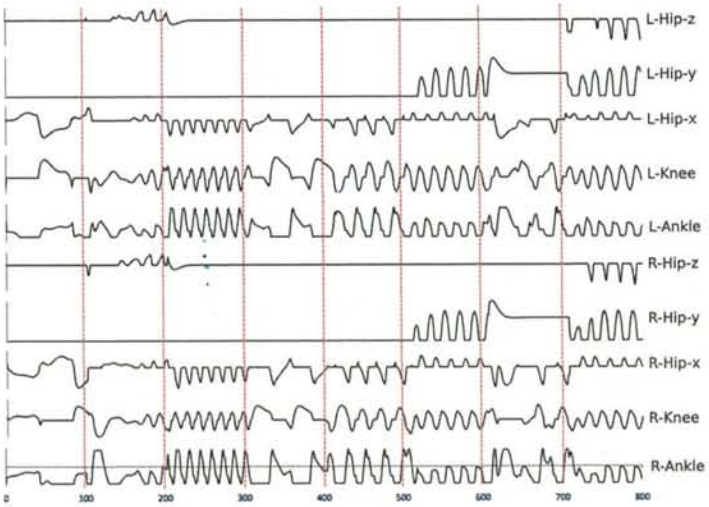



**Figure 5.60:** *Proposed MLP structure*




(a)




(b)

**Figure 5.61:** *Generated dynamic signal movement in different behavior (a) signal output of the motor neurons (b) the signal output converted in joint angle level*

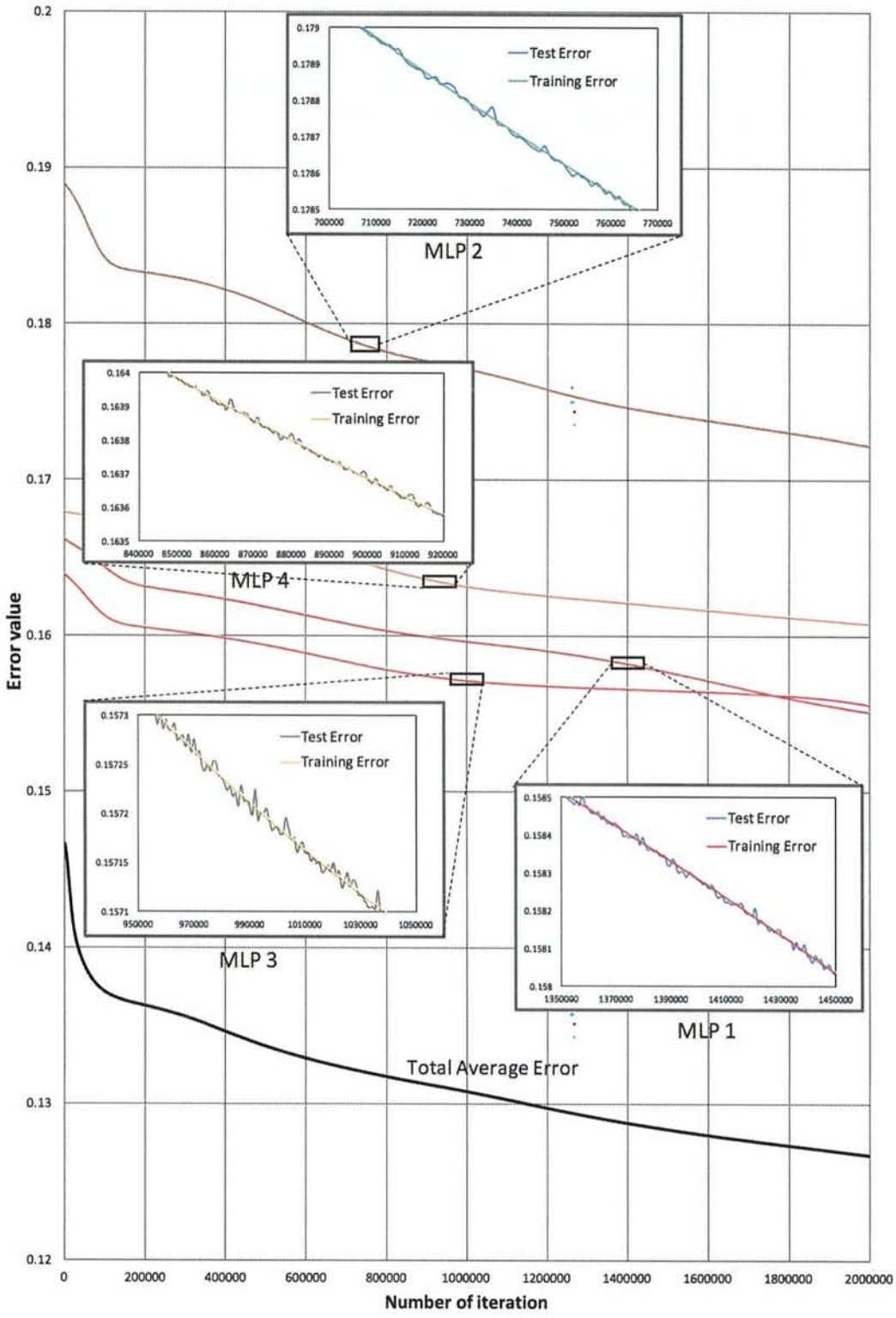



**Figure 5.62:** *Average error evolution with 4 samples of MLP's cross validation errors*

provision changing in every 100 time sampling and Fig.5.61b shows the dynamic signal in joint angle level. The sample of neuron interconnections can be seen in Fig. 5.63.

### 5.5.6 Conclusion and Discussion

This research proposed the learning strategy for solving the complexity of neural structure in neural oscillator based locomotion for generating omni-directional movement behavior. Since multi-objective evolutionary algorithm is used as the behavior optimization, movement provision and resulted energy are calculated as the fitness calculation, desired movement behavior can be acquired with minimum energy required. In this optimization, the behavior solutions was not deserved well, however, by increasing the number of generation, good diversity can be acquired. In this proposed method, walking behavior references generation took a long time, one reference behavior requires one optimization process. This is one of the problems which should be solved in next development.

Walking behavior references is used as the training data of the learning model. Separated MLP strategy is successfully approaching the relationship between input and output of walking behavior. Training iteration and learning process take a long time, and also require higher performance of learning model, such deep learning model. Input behavior references are also required to be increased in order to specify variant behavior in one movement provision, since in this current state, redundant behavior may be generated in the walking reference optimization. However, by using this proposed learning model with cross validation, omni-directional movement behavior can be generated. The model can generate sagittal movement from 0 m/s - 25 m/s, coronal movement from -5 m/s - 5 m/s, and turning speed $(-\pi/2)^0/s$ - $(\pi/2)^0/s$.

In the future, variant of walking input references should be increased by considering internal and external sensory information in order to classify redundant behavior. In order to improve the performance of learning model, high performance deep learning with online application may be implemented.

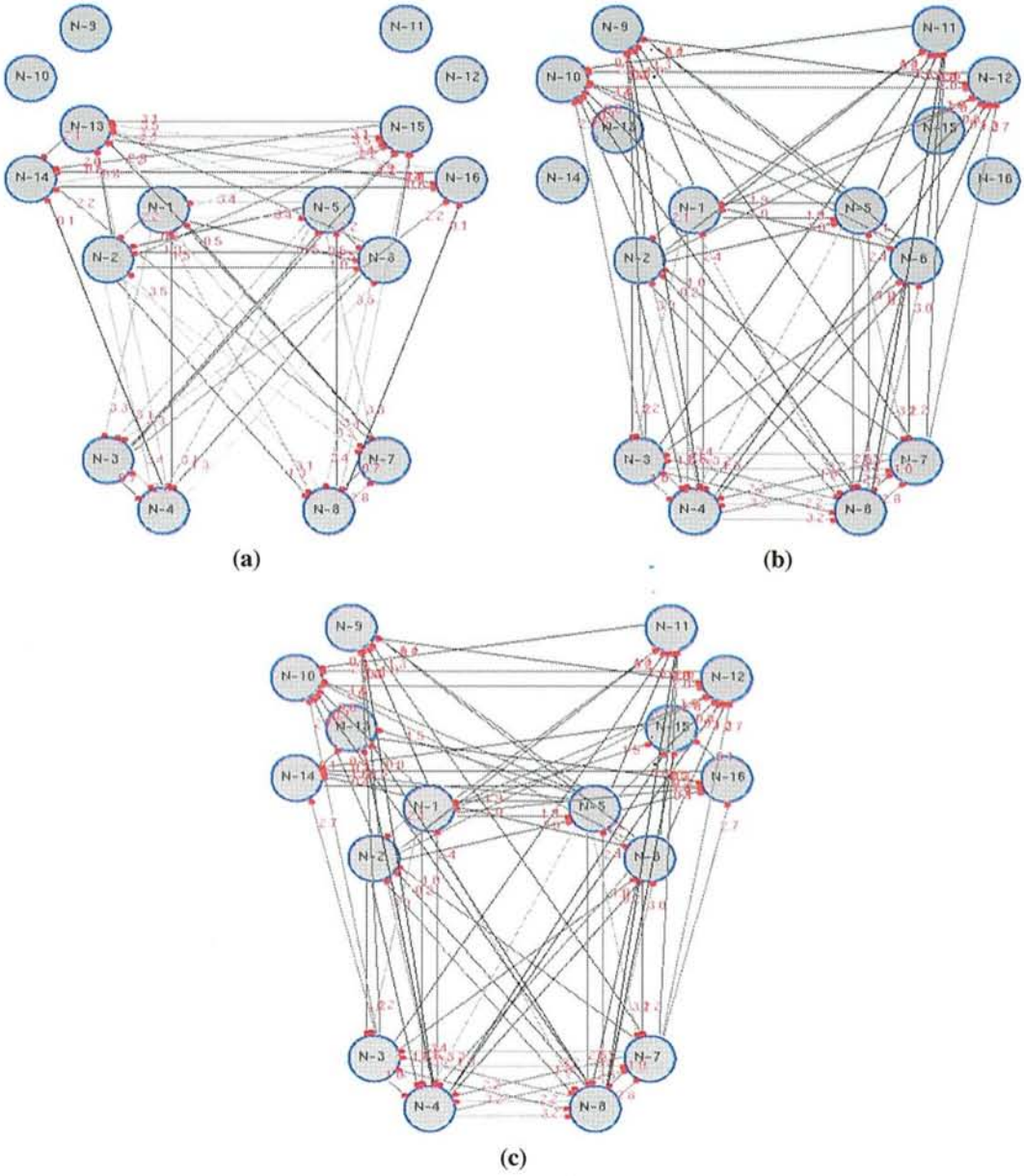



**Figure 5.63:** *Interconnection structure generated from different movement provision (a) structure of coronal walking behavior (b) structure of turning walking behavior (c) interconnection structure during combined coronal and turning provision*

# Chapter 6

# Stability Behavior

In order to support the locomotion behavior respond the disturbances coming, we developed stability model. We proposed biologically inspired stability model for avoiding the saturation phase and decresing robot's parameter requirements. There are several stability model developed in this proposed research which have been modified and evoluted from previous model in order to achieve desired stability behavior generation. In the first model, we used single modul recurrent neural network (RNN) as the learning model. In the first implementation, we use local effect strategy, Then, we implement dynamic synaptic strategy. After that, we design modular recurrent neural network which composed several RNN. In order to confirm the effectiveness of the proposed model, we test the model into two wheeled robot before testing in biped robot. Finally, we develop stability controller model for recovering external pertubation.

## 6.1 Local effect approach interconnection model

The stability system is built to have responsive ability to the disturber comes from environmental condition such as normal force resulted depending on surface condition. We designed the connection between sensoric neuron and joint neuron system that explained in Table 6.3. Local effect approach in this model was used to summarize the structure neuron. In these cases, we have three balancing systems: ground reaction, pose control, and hand reaction. Each of them has different connection structure.

### 6.1.1 Synapse Structure

In the proposed model, We implemented three different kind of sensors as mentioned in previous section. The synapse connection structure is illustrated in Fig. 6.1 and explained in

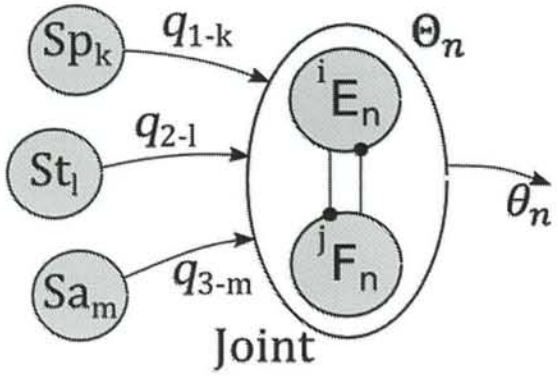



**Figure 6.1:** *Sensor connection structure in joint angle level*

Table 6.3. In ground reaction, there are four sensoric neurons located in the four corners of foot, $Sp_1$–$Sp_4$ for left foot and $Sp_5$–$Sp_8$ for right foot. These sensoric neurons are connected to ankle joint neurons: ankle-x joint and ankle-y joint. When front sensoric neurons ($Sp_1$, $Sp_2$, $Sp_5$, $Sp_6$) detect the ground, they send the positive impulse to ankle-x joint. When the right side sensoric neurons ($Sp_2$, $Sp_3$, $Sp_6$, $Sp_7$) detect the ground, positive impulse will be sent to joint ankle-y. These sensoric neurons (ground sensor) also influence knee joint and hip-x joint. When sensoric neuron detects the ground, it sends impulse to knee and hip-x joint. In tilt and angular velocity sensors, there are two degrees of freedom: pitch ($St_2$ and $Sa_2$) and roll ($St_1$ and $Sa_1$). $St_1$ and $Sa_1$ are connected to hand-x joint, while $St_2$ and $Sa_2$ are connected to ankle-y, hip-y, hand-y joint. The output sensors are resulted during the simulation process.

$$\theta_n = \Theta_n + \sum_{i=1}^{m} Sa_i q_{3,i} \alpha_{3,i}^n + \sum_{i=1}^{l} St_i q_{2,i} \alpha_{2,i}^n + \sum_{i=1}^{k} Sp_i q_{1,i} \alpha_{1,i}^n \qquad (6.1)$$

In Equation (6.1), $m$, $l$, $k$ are number of tilt, angular velocity, and ground sensor, respectively. The values of synapse $q_{1,i}$, $q_{2,i}$, and $q_{3,i}$ are controlled by using recurrent neural network (RNN) for acquiring the best stabilization parameter. This system will be detailed explained in the next section. $\theta_n$ represents the angle joint in $n$th joint. $\alpha_{3,i}^n$, $\alpha_{2,i}^n$, $\alpha_{1,i}^n$ represent the impuls effect of angular velocity sensor, tilt sensor, and ground sensor in n-th joint explained in Table 6.3, Since "0" implies there is no effect, "1" and "-1" imply positive and negative effect.

**Table 6.1:** *Connection structure*

| | Num. of neuron | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 | 32 |
| Tilt sensor $\left(s^{(t)}\right)$ | × | × | × | × | × | × | × | × | + | - | + | - | × | × | × | × | - | + | - | + | × | × | × | × | + | - | + | - | + | - | + | - |
| Ang. vel. sen. $\left(s^{(\omega)}\right)$ | × | × | × | × | × | × | × | × | + | - | + | - | × | × | × | × | - | + | - | + | × | × | × | × | + | - | - | + | + | - | - | + |
| Ground sen. 1-1 $\left(s_{1\text{-}1}^{(p)}\right)$ | - | + | + | - | + | - | - | + | - | - | - | - | - | + | + | - | + | - | - | - | × | × | × | × | × | × | × | × | × | × | × | × |
| Ground sen. 1-2 $\left(s_{1\text{-}2}^{(p)}\right)$ | - | + | + | - | + | - | - | + | - | - | - | - | - | + | + | - | - | + | + | - | × | × | × | × | × | × | × | × | × | × | × | × |
| Ground sen. 1-3 $\left(s_{1\text{-}3}^{(p)}\right)$ | - | + | + | - | + | - | - | + | - | - | - | - | - | + | + | - | - | + | + | - | × | × | × | × | × | × | × | × | × | × | × | × |
| Ground sen. 1-4 $\left(s_{1\text{-}4}^{(p)}\right)$ | - | + | + | - | + | - | - | + | - | - | - | - | + | - | - | + | - | + | + | - | × | × | × | × | × | × | × | × | × | × | × | × |
| Ground sen. 2-1 $\left(s_{2\text{-}1}^{(p)}\right)$ | + | - | - | + | - | + | + | - | - | - | - | - | + | - | - | + | - | + | + | - | × | × | × | × | × | × | × | × | × | × | × | × |
| Ground sen. 2-2 $\left(s_{2\text{-}2}^{(p)}\right)$ | + | - | - | + | - | + | + | - | - | - | - | - | + | - | - | + | + | - | - | + | × | × | × | × | × | × | × | × | × | × | × | × |
| Ground sen. 2-3 $\left(s_{2\text{-}3}^{(p)}\right)$ | + | - | - | + | - | + | + | - | - | - | - | - | + | - | - | + | + | - | - | + | × | × | × | × | × | × | × | × | × | × | × | × |
| Ground sen. 2-4 $\left(s_{2\text{-}4}^{(p)}\right)$ | + | - | - | + | - | + | + | - | - | - | - | - | - | + | + | - | + | - | - | + | × | × | × | × | × | × | × | × | × | × | × | × |

### 6.1.2 Design of RNN Model

In order to optimize the synapse weight, we utilized the recurrent neural network back propagation through time (BPTT) as depicted in Fig. 6.2. This system updates the correction synapse weight from sensoric neuron to joint neuron in real time. In our recurrent neural network system $p$ represents the state number of current condition of Robot. In this system we assumed the number of state condition is one. In Eq. (6.27), $x_i^p(t)$ is the input value in $t$ time sampling; $s_h^p(t-1)$ is output value of hidden neuron in $t-1$ time sampling; $b_j$ is bias input for hidden layer; $x_y^p(t)$ is output value of hidden neuron in t time sampling.

$$s_y^p(t) = f(ns_y^p(t)) = f\left(\sum_i^l x_i^p(t)\mathrm{V}_{ij}^p(t) + \sum_h^m s_h^p(t-1)\mathrm{U}_h^p + b_j\right) \tag{6.2}$$

$$y_k^p(t) = g(O_k^p(t)) = g\left(\sum_j^m s_j^p(t)W_{kj}^p(t) + b_k\right) \tag{6.3}$$

After forward process is done and resulted the output neuron value $y_k^p$ computed in Eq. (6.28), waiting time is required to acquire the sensor value $\mathrm{S}(k, y_k^p(t))$ and analyze the weight value. In Eqs. (6.29) and (6.30), $\delta_k^p$ is error propagation in output layer and is error propagation in hidden layer. We used hand actuator as the response output $y_k^p(t)$. The number of neurons in the input layer, in the hidden layer, and in the output layer are denoted by $l$, $m$, and $n$, respectively.

$$\delta_k^p = (d_k^p - \mathrm{S}(k, y_k^p(t)))g'(O_k^p(t)) \tag{6.4}$$

$$\delta_j^p = \sum_k^n \delta_k^p \mathrm{W}_{kj}^p(t) f'(ns_j^p(t)) \tag{6.5}$$

$$\mathbf{W}^p(t+1) = \mathbf{W}^p(t) + \eta_w \mathbf{s}^p(t)\delta_k^p \tag{6.6}$$

$$\mathbf{V}^p(t+1) = \mathbf{V}^p(t) + \eta_v \mathbf{x}^p(t)\delta_j^p \tag{6.7}$$

$$\mathbf{U}^p(t+1) = \mathbf{U}^p(t) + \eta_u \mathbf{s}^p(t-1)\delta_j^p \tag{6.8}$$

The activation function for hidden layer $f(x)$ used sigmoid function. $d_k^p$ is the desire output and the output function for are sigmoid function. In this case, the desire output is the condition where robot maintains the angular velocity of robot becomes zero. We have 12 neurons in the input layer resulted from sensors ($Sp_k$ ,$St_l$ ,$Sa_m$), four neurons in the hidden layer, and 12 neurons in the output layer. The configuration of input neuron is explained in Table 6.2. We need to train the parameters **V**, **U**, and **W** as the weight among neuron. $\eta_v$, $\eta_u$, and $\eta_w$ are the learning rate for **V**, **U**, **W** parameter recursively. In BPTT, the error propagation is done recursively. As depictied in Eqs. (7.10), (6.32), and (7.2), the weight of synapse between

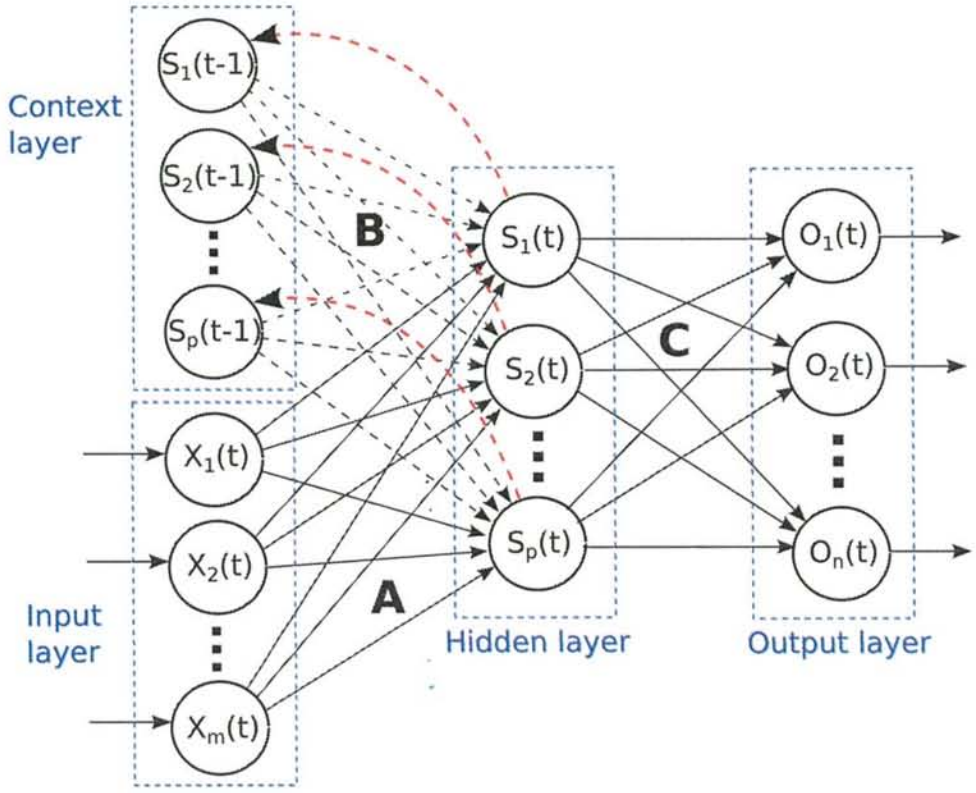



**Figure 6.2:** *Recurrent neural network structure*

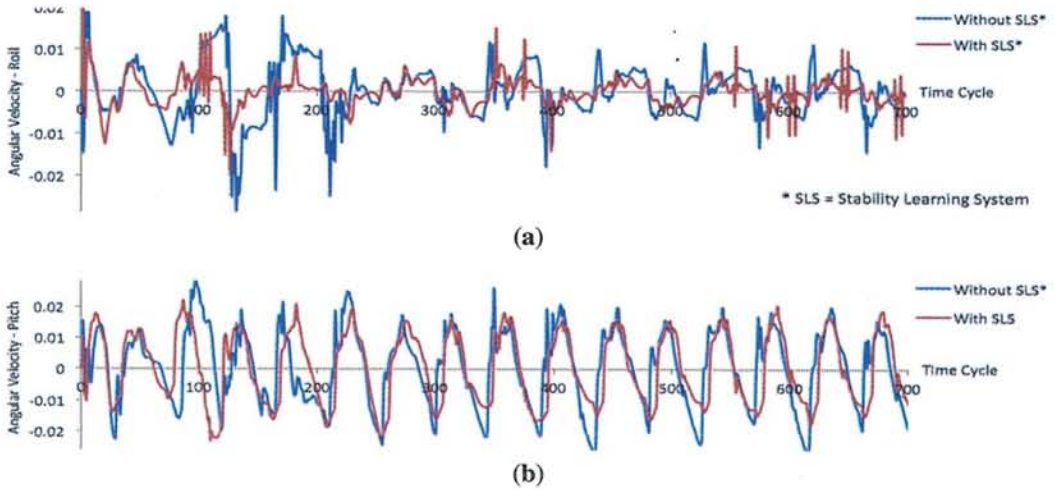



**Figure 6.3:** *Comparison angular velocity data between locomotion system without SLS and with SLS (a) Roll direction (b) Pitch direction*

sensoric and joint neuron can be regenerated.

**Table 6.2:** *Configuration of Input and Output Neuron*

| $x_i^p$ | $x_1^p$ | $x_2^p$ | $x_3^p$ | $x_4^p$ | $x_5^p$ | $x_6^p$ | $x_7^p$ | $x_8^p$ | $x_9^p$ | $x_{10}^p$ | $x_{11}^p$ | $x_{12}^p$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $S(k, y_k^p)$ | $S(y_1^p)$ | $S(y_2^p)$ | $S(y_3^p)$ | $S(y_4^p)$ | $S(y_5^p)$ | $S(y_6^p)$ | $S(y_7^p)$ | $S(y_8^p)$ | $S(y_9^p)$ | $S(y_{10}^p)$ | $S(y_{11}^p)$ | $S(y_{12}^p)$ |
| Input | $St_1$ | $St_2$ | $Sa_1$ | $Sa_2$ | $Sp_1$ | $Sp_2$ | $Sp_3$ | $Sp_4$ | $Sp_5$ | $Sp_6$ | $Sp_7$ | $Sp_8$ |
| Output | $q_{2,1}$ | $q_{2,2}$ | $q_{3,1}$ | $q_{3,2}$ | $q_{1,1}$ | $q_{1,2}$ | $q_{1,3}$ | $q_{1,4}$ | $q_{1,5}$ | $q_{1,6}$ | $q_{1,7}$ | $q_{1,8}$ |

### 6.1.3 Experimental Result

In this experiment, we continued to implement the stability system that used recurrent neural network model to support the balancing system. We defined $\eta_v = 0.02$, $\eta_u = 0.02$, and $\eta_w = 0.01$ as the learning rate. We analyzed the signal oscillation without stability learning system (SLS) and with SLS to compare the difference among them. Figure 6.3 and 6.4 shows thes angular velocity oscillation and tilt oscillation comparison in both of locomotion system. The locomotion with SLS has smaller tilt and angular velocity oscillation, which imply the locomotion with SLS more stable than locomotion without SLS. Beside that, the SLS also increase the speed of walking in robot locomotion as shown in Fig. 6.6. The locomotion with SLS can reach the length of movement longer because the SLS can effectively reduce the disturbance that hamper the robot walking. The weight between sensoric neuron and joint

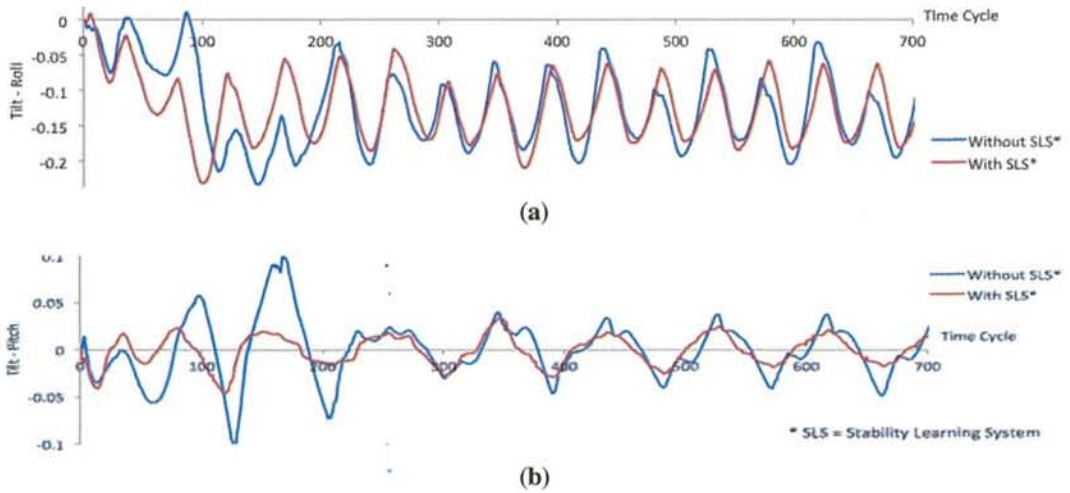



**Figure 6.4:** *Comparison angular velocity data between locomotion system without SLS and with SLS (a) Roll direction (b) Pitch direction*

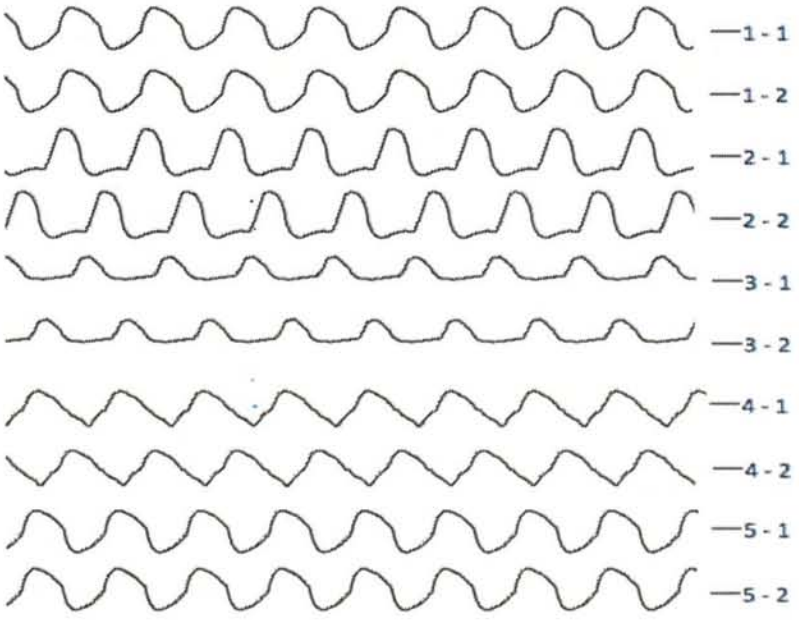



**Figure 6.5:** *Signal output of coupled neuron*

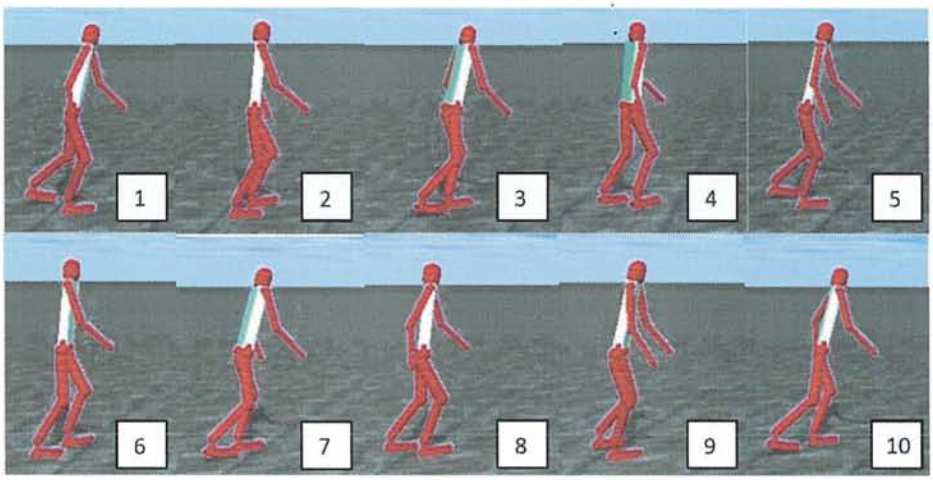



**Figure 6.6:** *Running process in ODE simulation*

neuron is changing every time sampling depending on the robot condition. SLS is effective for balancing system of humanoid robot locomotion. Beside that, the values of time sampling configuration are important. We have captured the running process simulation as seen in Fig. 6.5. This model is expected forming the human-like locomotion system.

The output of coupled neuron representing the signal of joints in robot when walking on flat surface was recorded as shown in Fig. 12. The hip-x, knee, and ankle-x joints have phase difference 180 degrees, while hip-y and ankle-y joints have the same phase. These signals have been influence with ground reaction. Therefore, these signals have different pattern depending on the environment condition. In this research, we only consider the disturbances in the flat surface. We proved that the stability system can reduce the oscillation of robot walking and also improve the speed of robot walking. This biological approach based robot locomotion are needed improvement, therefore the robot can walk in different direction, walk in uneven surface, and reduce the external disturbance.

### 6.1.4 Conclusion

In this research, we have designed the humanoid biped robot locomotion based on neural oscillator. The locomotion pattern in humanoid robot can be acquired by controlling the weight of synapse by using evolutionary algorithm. The design of neural structure between sensoric neuron and joint neuron could change the stabilization approach by using ZMP method. The stabilization of locomotion can be increased by learning the weight of synapse using recurrent neural network. Signal pattern resulted from the coupled neuron is suitable for the ground reaction. We have designed the synapse from sensoric to joint neuron and the

walking movement without turning left or right. As the future research, we are designing to add a neuron systems adapted as brain neuron systems that process the information from sensoric neuron system and generate signal to motoric neuron system. Brain neuron systems also control the timing impulse for controlling the signal generated by motoric neuron.

## 6.2 Dynamic Synaptic Value for Walking Stability

In this stability learning system, we used a learning system based on a biological approach in order to realize adaptive walking, while [117] and [118] used a conventional method, such as center of gravity (CoG), in order to realize adaptive walking in both biped robot and quadruped robot. The proposed system is responsive to disturbances from inside and outside the system. The value from outside system disturbance, such as normal force, varies, depending on the surface condition. In this section, we design the structure of interconnection between the sensoric neuron and the motoric neuron (Fig. 5.1). In order to simplify the structure of neurons, the local effect approach was utilized in this model. The local effect approach implies that the sensoric neuron will affect the motoric neurons surrounding it. The touch sensor in the soles of the feet influenced the motoric neuron to actuate the muscle in the ankle area.

The proposed stability system consists of three balancing systems (ground reaction, pose control, and hand reaction) with differing neuron structure.

### 6.2.1 Synapse Structure between Sensoric and Motoric Neurons

In this model, as explained in the previous section, there are three kinds of sensor. For ground reaction, there are four sensoric neurons located in the corners of the feet (Fig. 6.7b). These sensoric neurons are connected to the motoric neuron in both ankle-x joint and ankle-y joint. When two sensoric neurons ($s_3^{(p)}$ and $s_4^{(p)}$) detect the ground, they send a positive impulse to the flexor neuron in ankle-x joint and send a negative impulse to the extensor neuron in ankle-x joint. On the other hand, when the left side sensoric neurons ($s_1^{(p)}$ and $s_4^{(p)}$) detect the ground, a positive impulse is sent to the flexor neuron and a negative impulse is sent to the extensor neuron in ankle-y joint. As shown in Fig. 6.7c, the sensoric neurons ($s_k^{(p)}$) also influence the knee joint and the hip-x joint. When the sensoric neuron detects the ground, it sends a positive impulse to the extensor neuron and a negative impulse to the flexor neuron in knee and hip-x joint. The tilt and angular velocity sensors have two degrees of freedom: pitch and roll. The sensoric neuron for roll direction is connected to the motoric neuron in the hand-x joint. Meanwhile, the sensoric neuron for pitch direction is connected

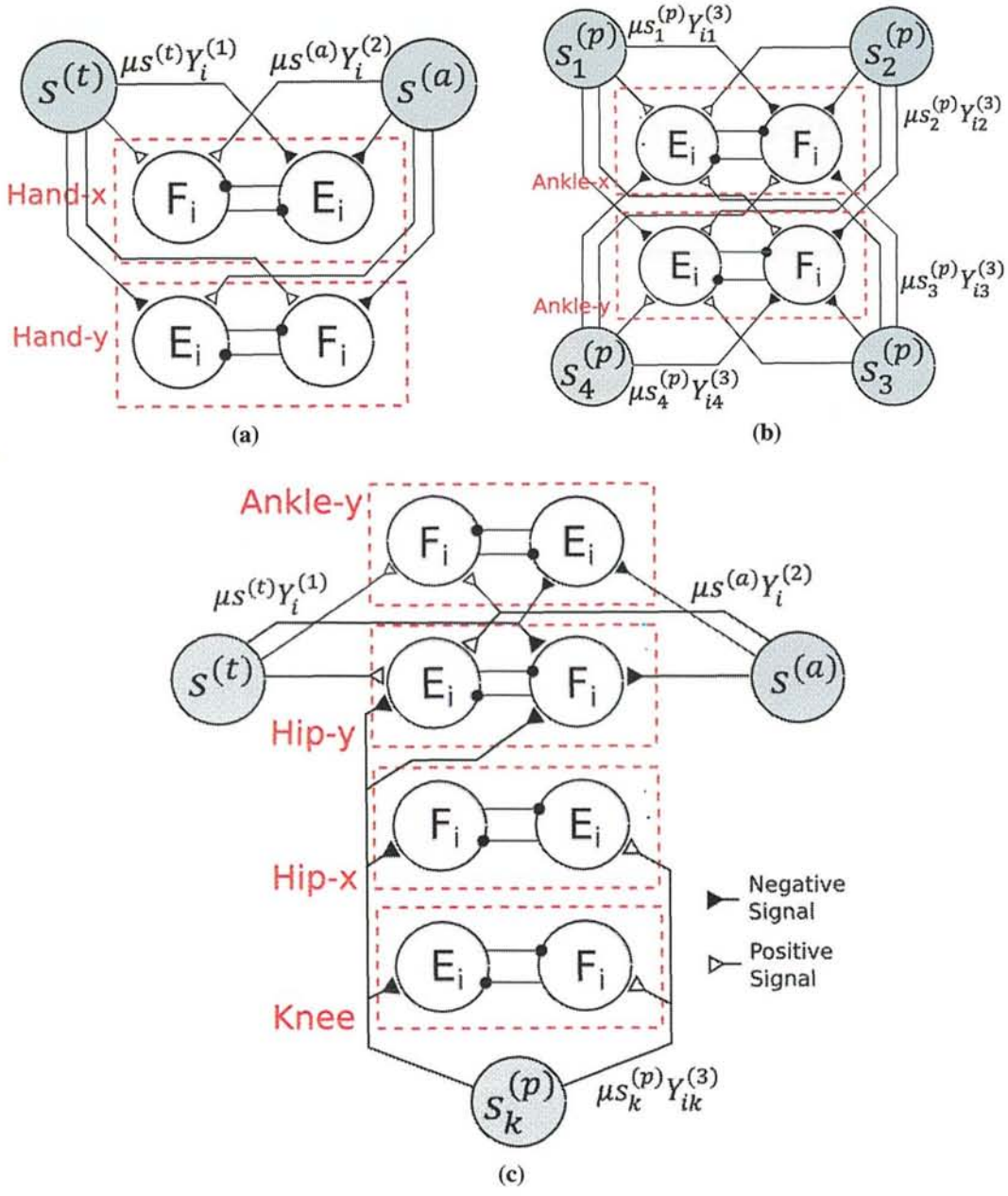



**Figure 6.7:** *a) Sensor connection in hand area; b) Sensor connection in soles of feet; c) Sensor connection in foot*

**Table 6.3:** *Sensoric and Motoric Connection*

| | Num. of neuron | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 | 32 |
| Tilt sensor $(s^{(t)})$ | × | × | × | × | × | × | × | × | + | - | + | - | × | × | × | × | - | + | - | + | × | × | × | × | + | - | + | - | + | - | + | - |
| Ang. vel. sen. $(s^{(a)})$ | × | × | × | × | × | × | × | × | + | - | + | - | × | × | × | × | - | + | - | + | × | × | × | × | + | - | - | + | + | - | - | + |
| Ground sen. 1-1 $(s^{(g)}_{1,1})$ | - | + | + | - | + | - | - | + | - | - | - | - | - | + | + | - | + | - | - | - | × | × | × | × | × | × | × | × | × | × | × | × |
| Ground sen. 1-2 $(s^{(g)}_{1,2})$ | - | + | + | - | + | - | - | + | - | - | - | - | - | + | + | - | - | + | + | - | × | × | × | × | × | × | × | × | × | × | × | × |
| Ground sen. 1-3 $(s^{(g)}_{1,3})$ | - | + | + | - | + | - | - | + | - | - | - | - | - | + | + | - | - | + | + | - | × | × | × | × | × | × | × | × | × | × | × | × |
| Ground sen. 1-4 $(s^{(g)}_{1,4})$ | - | + | + | - | + | - | - | + | - | - | - | - | + | - | - | + | - | + | + | - | × | × | × | × | × | × | × | × | × | × | × | × |
| Ground sen. 2-1 $(s^{(g)}_{2,1})$ | + | - | - | + | - | + | + | - | - | - | - | - | + | - | - | + | - | + | + | - | × | × | × | × | × | × | × | × | × | × | × | × |
| Ground sen. 2-2 $(s^{(g)}_{2,2})$ | + | - | - | + | - | + | + | - | - | - | - | - | + | - | - | + | + | - | - | + | × | × | × | × | × | × | × | × | × | × | × | × |
| Ground sen. 2-3 $(s^{(g)}_{2,3})$ | + | - | - | + | - | + | + | - | - | - | - | - | + | - | - | + | + | - | - | + | × | × | × | × | × | × | × | × | × | × | × | × |
| Ground sen. 2-4 $(s^{(g)}_{2,4})$ | + | - | - | + | - | + | + | - | - | - | - | - | - | + | + | - | + | - | - | + | × | × | × | × | × | × | × | × | × | × | × | × |

to the motoric neuron in the ankle-y, hip-y, and hand-y joints. The inter-connections between the sensoric neuron and motoric neurons in Fig. 6.7 are designed based on some preliminary tests. The detailed connection between the sensoric neuron and the motoric neuron is shown in Table 6.3, where the notation "x" means that there is no connection; "-" means that there is a negative effect of the sensoric neuron in relation to the motoric neuron; "+" means that there is a positive effect of the sensoric neuron in relation to the motoric neuron. The changing of synapse values $\mathbf{Y}^{(1)}$, $\mathbf{Y}^{(2)}$, $\mathbf{Y}^{(3)}$ is controlled by RNN in order to acquire the best stabilization parameter. This system will be detailed in the next subsection.

### 6.2.2 Structure Model of Recurrent Neural Network (RNN)

Many researchers, such as [203], [194], [59], [60], and [115], have used RNN to support their locomotion model. Tran et al. used RNN as the oscillator to generate the periodic signal and CPG for controlling the walking mechanism [203]. RNN was also applied for generating a suitable behavior sequence for an autonomous robot in [194]. While other researchers have used RNN for stabilization in biped robot locomotion, Fukuda et al. combine RNN with self-adaptive GAs to achieve learning capability [59], [60]. In 2015, Lin et al. used RNN to control biped robot locomotion [115]. However, one researcher has used neural network (NN) for the controller system [180].

In order to learn the synapse weight between the sensoric and the motoric neuron, the RNN Elman model with back propagation through time (BPTT) is used (Fig. 6.2). Figure 6.8 shows the transferring process of the weight to the synapse that connects the sensoric neuron to the motoric neuron. The RNN system transfers a different weight value in each time cycle, depending on the condition of the feedback sensor and the actuator. This system updates the correction synapse weight value between the sensoric neuron and the motoric neuron in real time.

In this research, the RNN system is divided into three subsystems: ground reaction, pose

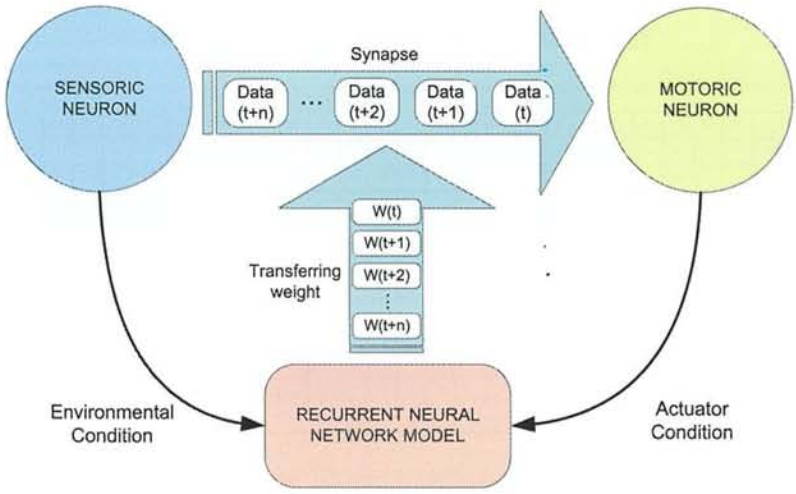



**Figure 6.8:** *Process of weight transfer*

control, and hand reaction. $\delta_k$ is the error propagation in the output node and $\delta_j$ is the error propagation in the hidden node computed in Eqs. (6.29) and (6.30), where $d_k$ is the desired output and the output function $g(x)$ is the output of the sensor. In this case, the desired output is the condition where the robot maintains the angular velocity at zero value. In Eq. (6.27) and (6.28), $X_i(t)$ is the input value in certain time from the sensor feedback from the $i$th input neuron, $S'_j(t)$ and $S_j(t)$ are the inputs and outputs of the $j$th hidden neuron, and $O'_k(t)$ and $O_k(t)$ are the inputs and outputs of the $k$th output neuron. The number of neurons in the input layer, in the hidden layer, and in the output layer are denoted by $m$, $p$, and $n$, respectively.

$$S_j(t) = f(S'_j(t)) = f\left(\sum_i^m X_i(t)A_{ij}(t) + \sum_h^p S_h(t-1)B_{hj}(t)\right) \tag{6.9}$$

$$O_k(t) = f(O'_k(t)) = f\left(\sum_j^p S_j(t)C_{jk}(t)\right) \tag{6.10}$$

$$\delta_k = (d_k - g(O_k(t)))f'(O'_k(t)) \tag{6.11}$$

$$\delta_j = \sum_k^n \delta_k C_{jk}(t) f'(S'_j(t)) \tag{6.12}$$

$$\mathbf{C}(t+1) = \mathbf{C}(t) + \eta \mathbf{S}(t)\delta_k \tag{6.13}$$

$$\mathbf{B}(t+1) = \mathbf{B}(t) + \eta \mathbf{X}(t)\delta_j \tag{6.14}$$

$$\mathbf{A}(t+1) = \mathbf{A}(t) + \eta \mathbf{S}(t-1)\delta_j \tag{6.15}$$

In Eq. (6.27), the activation function for the hidden layer, $f(x)$ uses the sigmoid function.

In Equations (7.10), (6.32), and (7.2), the weight parameters of the neurons are presented by **A**, **B**, and **C** matrices acquired by a learning process. In BPTT, the error propagation is done recursively, where $\eta$ is the learning rate of the weight of synapse between the motoric and the sensoric neuron. These weight parameters can be dynamically regenerated depending on the environmental condition.

**Table 6.4:** *Clustering Sensoric Weights*

| Output RNN | Connection parameter |
|---|---|
| $O_1$ | $Y_1^1 \rightarrow$ Hand |
| $O_2$ | $Y_2^1 \rightarrow$ Ankle-y |
| $O_3$ | $Y_3^1 \rightarrow$ Hip-y |
| $O_4$ | $Y_1^2 \rightarrow$ Hand |
| $O_5$ | $Y_2^2 \rightarrow$ Ankle-y |
| $O_6$ | $Y_3^2 \rightarrow$ Hip-y |
| $O_7$ | $Y_1^3 \rightarrow$ Ankle-x, Ankle-y |
| $O_8$ | $Y_2^3 \rightarrow$ Knee |
| $O_9$ | $Y_3^3 \rightarrow$ Hip-x |
| $O_{10}$ | $Y_4^3 \rightarrow$ Hip-y |

In this recurrent model, we used 10 input RNN neurons as the control parameter: tilt sensor, angular velocity sensor, and eight ground sensors. The hidden layer consists of 16 neurons, while in the output layer there are 10 neurons representing the weight of the signal effect from the sensoric neuron. The input value and output value are normalized from 0 to 1. The detailed representation of the output neuron in the RNN model of inter-connection between the sensoric neuron and the motoric neuron can be seen in Table 6.4 and Fig. 6.7. Since the range of the output value $O_k$ is between -3 and 3, normalization of the value from 0 to 1 is needed to simplify the calculation.

### 6.2.3 Experimental Result

In this experiment, we utilized the system with stability system and without stability system, in which the value of gain is defined as follows: ($\mu_{gain} = \{0.6, 0.7, 0.8, 0.9, 1.0\}$). We set the learning rate $\eta$ as 0.02 and used parameters from the fourth solution ($W_4$). When we set the robot to low speed, there was no difference in stability level between the locomotion with and without the stability system. Therefore, we used locomotion with both high speed and long step walking.

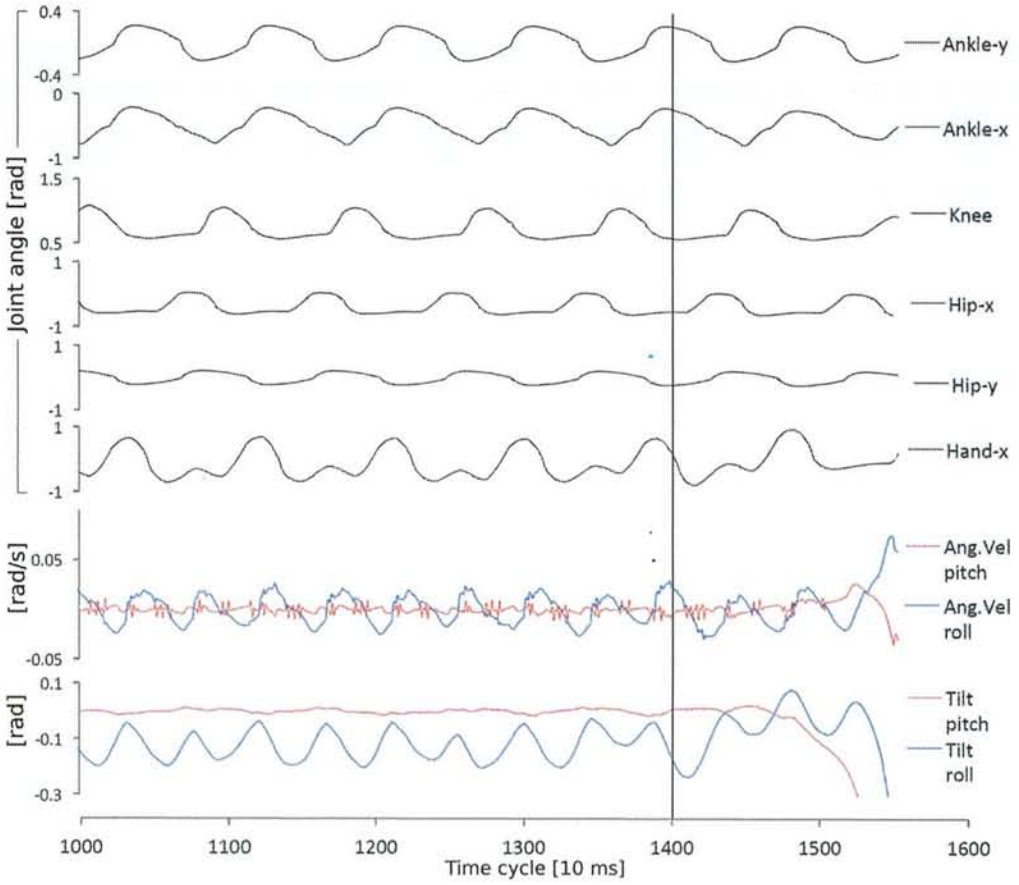



**Figure 6.9:** *The changes in joint angle signal, angular velocity, and body tilt angle from using stability system to without using stability system*

If we use only the locomotion pattern resulting from multi-objective optimization, without the stability learning system (feedback sensor), then the maximum value of $(\mu_{gain})$ that can be supported is 1. Therefore, in order to prove the effectiveness of the stability learning system, we set the gain parameter to 1.2. If the stability learning system can support the robot walking, then, this system can improve the stability of the robot walking.

First, we run the robot in simulation with the stability system. After 1400 time samplings, we stop using the stability system: this results in the robot falling down at 1560 time samplings (Fig. 6.9). The other experimental result comparison between locomotion with and without stability system can be seen in Table 6.5.

In this experiment, we also measured the zero moment point (ZMP) trajectory in y-direction and x-direction for robot without stability system (Fig. 6.10) and robot with stability system (Fig. 6.11). The ZMP trajectories in Fig. 6.11 have smaller ripple and more stability than the ZMP trajectories without the stability system. However, both ZMP trajectories always lie inside the supported area. When compared with existing research, the stability

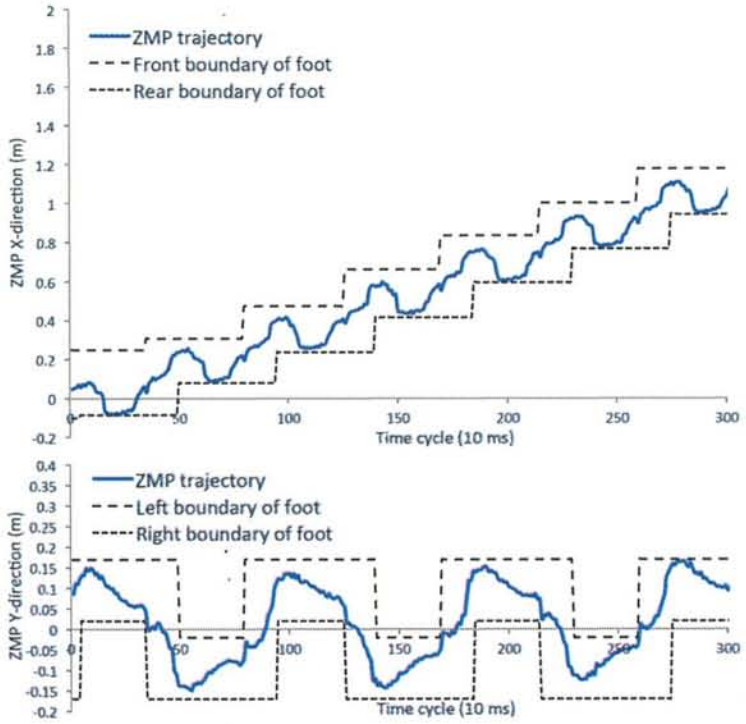



**Figure 6.10:** *ZMP trajectory during straight walking without stability learning system. Gain parameter* $(\mu_{gain}) = 1.0$ *and time cycle = 0.01*

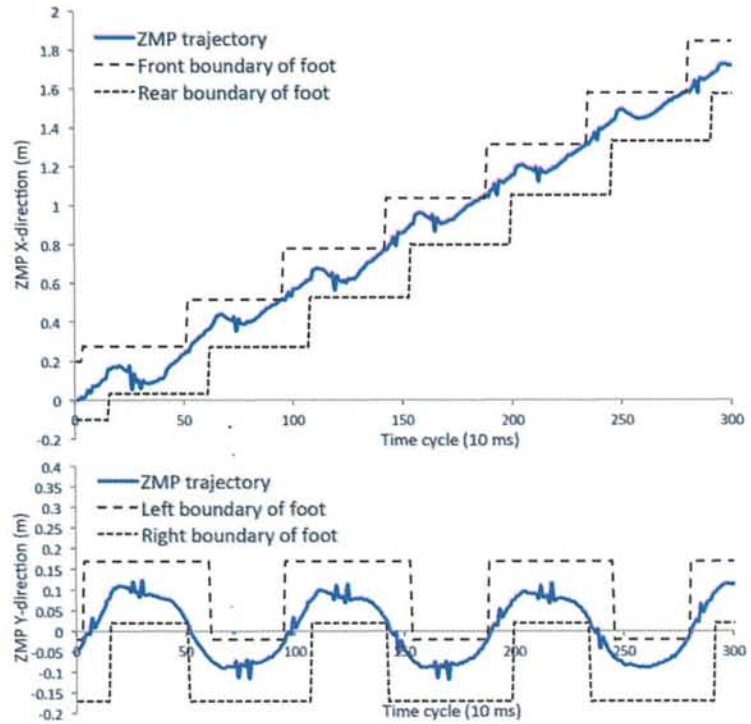



**Figure 6.11:** *ZMP trajectory during straight walking by using stability learning system with gain parameter* $(\mu_{gain}) = 1.2$ *and time cycle = 0.01*

**Table 6.5:** *Comparison With Stability System and Without Stability System*

| Gain ($\mu_{gain}$) | Without Stability System | | With Stability System | |
|---|---|---|---|---|
| | Condition | Velocity | Condition | Velocity |
| 0.6 | OK | 0.225 m/s | OK | 0.240 m/s |
| 0.7 | OK | 0.249 m/s | OK | 0.270 m/s |
| 0.8 | OK | 0.330 m/s | OK | 0.325 m/s |
| 0.9 | OK | 0.375 m/s | OK | 0.373 m/s |
| 1.0 | OK | 0.425 m/s | OK | 0.429 m/s |
| 1.2 | FALL | 0 m/s | OK | 0.56 m/s |

system proposed in this research reduces the ripple of ZMP; therefore, this locomotion has a smaller ripple than that of existing studies [148], [79], [149]. Moreover, our locomotion has a higher ratio between the length of step and the length of foot, and has a shorter time for the double support phase (DSP) than that shown in other research.

### 6.2.4 Stability analysis and Discussion

In order to analyze the stability of robot locomotion during walking, we recorded the inertial data of the robot when walking for 2000 time cycles. The recorded data were then presented using the phase diagram from the tilt body of the robot in roll direction, based on the Poincare map, as depicted in Fig. 6.12. Symbol $p_n$ (Poincare point) in Figs 6.12 and 6.13 is the point value of tilt angle of the robot body, where the angular velocity is zero and the previous angular velocity is greater than zero. The initial condition of tilt is 0 rad; it gradually changed to a stable tilt angle of 0.12 rad. We also analyzed the behavior and the stability of locomotion using a Cobweb diagram. The Cobweb diagram shows the alteration of $p_n$ from the initial point to the stable point. Therefore, this locomotion system is stable.

In comparison with existing locomotion systems, our proposed locomotion system can reach stability faster, according to the Poincaré map and Cobweb diagram presented in Figs 6.12 and 6.13. In the proposed system, the stability point was reached in only five time cycles from the starting point, while in [142], the stability point was reached after eight time cycles from the starting point. In addition, since the proposed system has an independent system to support stabilization, the stability system of neural-based locomotion in our system is better when compared with existing methods. Moreover, in order to support the feedback system, the proposed system was also equipped with additional sensors (as mentioned in the previous section).

In conclusion, this experiment proved that the stability system in this research is used effectively for locomotion based on a biological approach. We developed the stability system

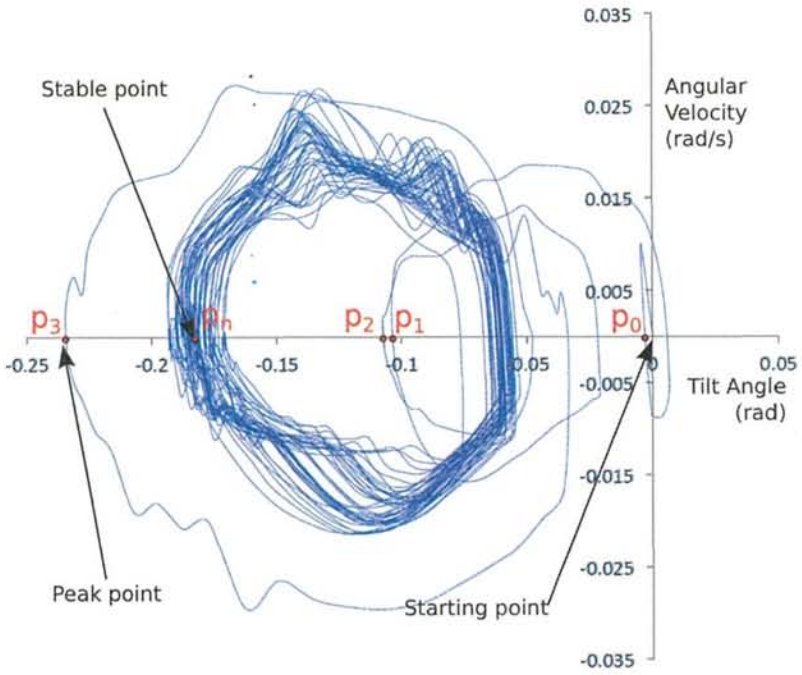



**Figure 6.12:** *Phase diagram of robot tilt angle and stability analysis, based on Poincare map*

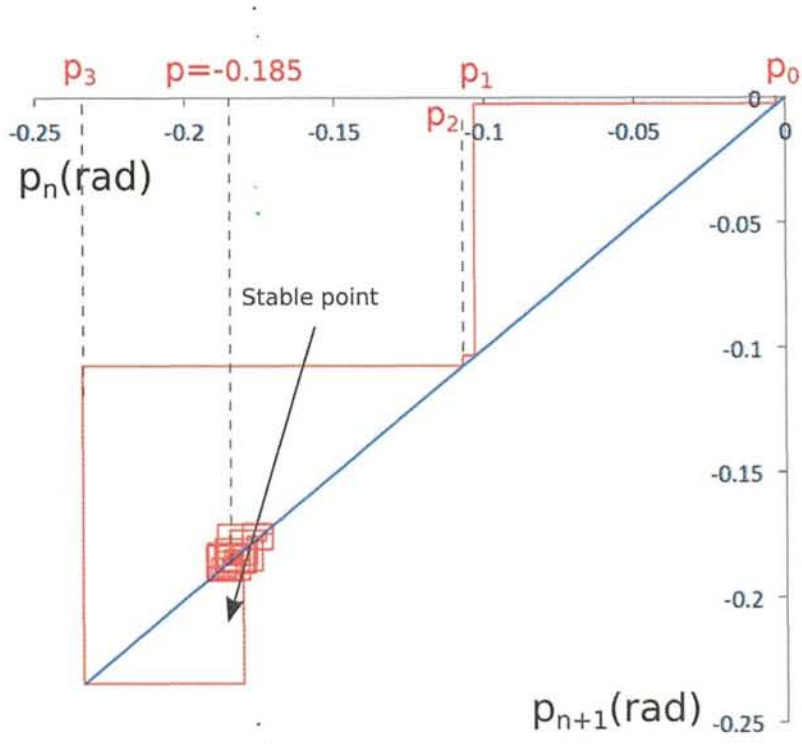



**Figure 6.13:** *Cobweb diagram representation of Figure 6.12*

and the locomotion system separately. This stabilization approach could maintain the stability of the robot with the learning system. In order to emphasize our stabilization system, we implemented the simulation result with a real robot, as explained in the next subsection.

## 6.3 Design of Multimodal Recurrent Neural Network (MRNN)

This research proposed the new design of multimodal neural neural network inspired from human learning system which takes different action in different condition. The multimodal neural network consists of some recurrent neural networks (RNNs) those are separated into different condition. There is selector system that decides certain RNN system depending the current condition of the robot. In this research, we implemented this system in pendulum mobile robot as the basic object of study. Several certain number of RNNs are implemented into certain different condition of tilt robot. RNN works alternately depending on the condition of robot. In order to prove the effectiveness of the proposed model, we simulated in the computer simulation Open Dynamics Engine (ODE) and compared with ordinary RNN. The proposed neural model successfully stabilize the applied robot (2-wheeled robot). This model is developed for implemented into humanoid balancing learning system as the final object of study.

### 6.3.1 Introduction

There has been a significant increase in the development of robotic stability systems. Most researchers implemented physical method for the stabilization. In our previous research, we also implemented physical approach. We applied inverted pendulum model and zero moment point in humanoid robot locomotion [176], [169]. Other researcher implemented biological approach learning system for stability system or control system in any object of studies. In biological approach, recurrent neural networks are suitable for control system and have been applied in several cases. Lin et al. proposed a robust recurrent neural network (RRNN) for a biaxial motion mechanism in CNC machine. The aim of their proposed method is for improving the motion tracking performance [116]. In 2008, Zhang et al. implemented RNN for model predictive control. Zhang et al. shows the effectiveness of their proposed method in simulation [236]. Other researchers implemented diagonal RNN that is known for dynamic mapping and for nonlinear dynamical systems. It also can be combined with conventional PID controller for tracking a straight line under the trapezoidal velocity

planning [112].

In a specific case, especially in humanoid robot locomotion, there are many researcher also implemented RNN in humanoid robot that will be final object of study in our proposed research. In 2014, Tran et al. developed central pattern generation based the biped robot locomotion and applied RNN controlling the walking mechanism and walking stabilization in their applied robot [203]. In the same year, Jeong et al applied RNN for for generating the suitable behavior sequence for autonomous robot [194]. In 2015, Lin et al. implemented RNN with Elman model to achieve satisfactory control without performance degradation for humanoid biped robot locomotion with unknown uncertainties and faults [115]. RNN also can be combined with evolutionary algorithm for controlling the stability of humanoid robot [59].

In our previous research, we implemented single RNN Elman model with back propagation through time (BPPT) for stability system in humanoid robot locomotion [168]. In this proposed research, we will improve the stability level in robotics cases. This research are inspired from the human learning system. Humans give different responses in different condition. Therefore, we applied multimodal recurrent neural network composed by some RNNs system which one RNN represent one certain condition. In this proposed method we used RNN Elman model [35]. In modular neuron structure studies, Yamaguchi et al. proposed a new modular neural network architecture used simple feedback controller for training the neural networks in each module. The proposed research was applied for a mobile robot controller [223]. In 2000, modular learning system has been proposed by Farooq. Different with our proposed model which considers previous condition, he implemented ordinary artificial neural network in each module without considered previous condition [16]. Multimodal neural system has been proposed by several researcher. Kiros et al. proposed multimodal neural language model for image processing [98]. Other researchers also implemented multimodal neural learning system for image processing [213]. Modular Neural Network is also implemented for adaptive control in Arm robot manipulator. It's involving Support Vector Machines (SVM) and an adaptive unsupervised neural network. Karras also implemented it for forming the trajectory mapping and tracking control [94].

We are going to implement this proposed learning model in stability system on humanoid robot in order to build a self-adaptation system. Before implementing in a humanoid robot as the final object of study and in order to prove the effectiveness of the proposed learning model, we implemented this learning model in two-wheeled mobile robot as the first object of study because its simplicity. In experimental result, we will compare our proposed method with conventional RNN which considers the previous condition to be processed. The contribution of this proposed model is that, we design multimodal learning system for stabil-

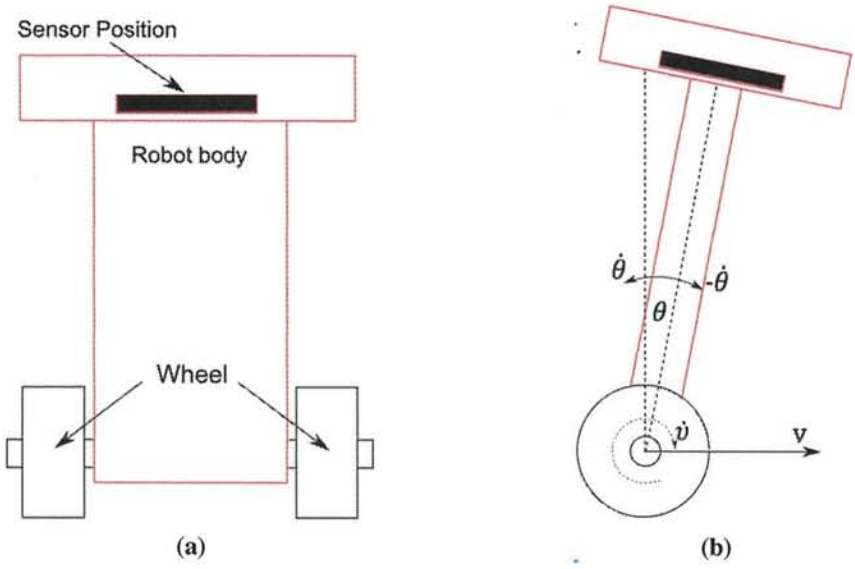



**Figure 6.14:** *a) Robot design from front side. b) Robot design from side*

ity which implement combined RNNs work alternately depending on the current condition. Therefore, by using this method robot can respond different action in different condition.

### 6.3.2 Overview Implemented Robot Design

In this section, we explain the robot model that we used to prove the effectiveness of the proposed model. We used two-wheeled mobile robot that is inspired from inverted pendulum model that is depicted in Fig. 6.14. This mobile robot model is often implemented for Segway, which is used for helping human move from one place to other place. This robot design is the simple implementation of the proposed model that can represent the balancing system in humanoid robot. We design the applied robot in the Open Dynamics Engine simulation. We installed 2 kinds of sensor in this robot, tilt sensor and angular velocity sensor. Those are represented the angle value of robot and angle speed value of robot from vertical direction, respectively.

In the future, we would like to implement the proposed model in the humanoid biped robot. Humanoid biped robot design represented by the applied robot design can be seen in Fig. 6.15. Our proposed robot design in this research has two wheels as the response actuators of the stabilization system. The wheels movement results the speed $v$ which is computed in Eq. (6.16). The speed value is depending the value of the output of certain $y$th RNN ($O_k^y(t)$) where it represents the speed acceleration of wheels. Each RNN module result different

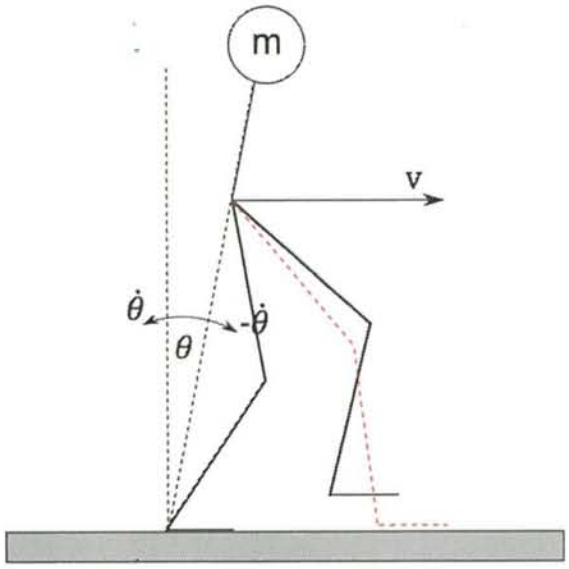



**Figure 6.15:** *Humanoid biped robot design representation of proposed robot design. The body speed is controlled by controlling the walking step behavior in further model*

speed acceleration of wheels depending on the sensors condition. In Eq. 6.16, $a_{min}$ and $a_{max}$ are the minimum and maximum value of the speed acceleration, respectively. Where, $\theta$ is the tilt angle of the robot from vertical direction and $\dot{\theta}$ is the angular velocity or the robot. In humanoid biped robot design depicted in Fig. 6.15, movement speed $v$ is resulted by step length of walking robot. However the additional modeling is required in order to model the walking behavior so that can control the speed of the robot walking. In the previous research, we have built the walking speed control [175] that will be combination with proposed research.

$$v(t) = v(t-1) + \left(O_k^y(t)\left(a_{max} - a_{min}\right) + a_{min}\right) \tag{6.16}$$

### 6.3.3 RNN model

In this section, we explain how MRNN's system implemented for two-wheeled mobile robot. Many researcher have used RNN model for control system. In the previous research, we used single RNN in order to build the stability system in humanoid robot [168]. In this proposed model, MRNN composed more than one RNN system which works alternately depending on the certain condition in the certain time. RNNs process their previous condition in current process, therefore the previous condition will influence and become input in the feed forward process.

In this research the RNN system is divided into four layers which are input layer, hidden

layer, context layer, and output layer. The number of hidden and context layer are same since context layer represents the previous condition of hidden layer. The mathematical model of feed forward process of RNN can be seen in Eq. (6.27) and (6.28). In Eq. (6.27) and (6.28), $X_i(t)$ is the input value in certain time from the sensor feedback from the $i$th input neuron, $S'_j(t)$ and $S_j(t)$ are the inputs and outputs of the $j$th hidden neuron, and $O'_k(t)$ and $O_k(t)$ are the inputs and outputs of the $k$th output neuron. The number of neurons in the input layer, in the hidden layer, and in the output layer are denoted by $m$, $p$, and $n$, respectively. Parameter $y$ shows the current RNN activated.

$$e = \begin{cases} 0 & \text{if } \alpha_t < (\alpha_{t-1} - \beta) \\ \text{sgn}\big(d_k - g(O_k(t))\big)\big(\alpha_t - (\alpha_t - \beta)\big) & \text{otherwise} \end{cases} \tag{6.17}$$

In this case, the desired output is the condition where the robot maintains the angular velocity at zero value. $\delta_k$ is the error propagation in the output node and $\delta_j$ is the error propagation in the hidden node computed in Eqs. (6.29) and (6.30), $e$ is the error value calculated in Eq. 6.26, where $\alpha = \text{abs}(d_k - g(O_k(t)))$, $d_k$ is the desired output, and the output function $g(x)$ is the output of the sensor.

$$S_j^y(t) = f(S'_j(t)) = f\left(\sum_i^m X_i(t)A_{ij}^y(t) + \sum_h^p S_h(t-1)B_{hj}^y(t)\right) \tag{6.18}$$

$$O_k^y(t) = f(O'_k(t)) = f\left(\sum_j^p S_j(t)C_{jk}^y(t)\right) \tag{6.19}$$

$$\delta_k^y = (e_t)f'(O'_k(t)) \tag{6.20}$$

$$\delta_j^y = \sum_k^n \delta_k^y C_{jk}^y(t) f'\big(S_j'^y(t)\big) \tag{6.21}$$

$$\mathbf{C}^y(t+1) = \mathbf{C}^y(t) + \eta \mathbf{S}^y(t)\delta_k^y \tag{6.22}$$

$$\mathbf{B}^y(t+1) = \mathbf{B}^y(t) + \eta \mathbf{X}^y(t)\delta_j^y \tag{6.23}$$

$$\mathbf{A}^y(t+1) = \mathbf{A}^y(t) + \eta \mathbf{S}^y(t-1)\delta_j^y \tag{6.24}$$

In Eq. (6.27), the activation function for the hidden layer, $f(x)$ uses sigmoid function. In Equations (7.10), (6.32), and (7.2), the weight parameters of the neurons are presented by **A**, **B**, and **C** matrices acquired by a learning process. In BPTT, the error propagation is done recursively, where $\eta$ is the learning rate of the weight of synapse between the motoric and the sensoric neuron. Those weight parameters can be dynamically regenerated depending on

the environmental condition.

In this RNN model, 2 neurons are used in input layer as the tilt or angular velocity angle and the previous speed acceleration of the wheels. Hidden layer and context layer consist 10 neurons, while in output layer consist 1 neuron represent the speed acceleration of the robot.

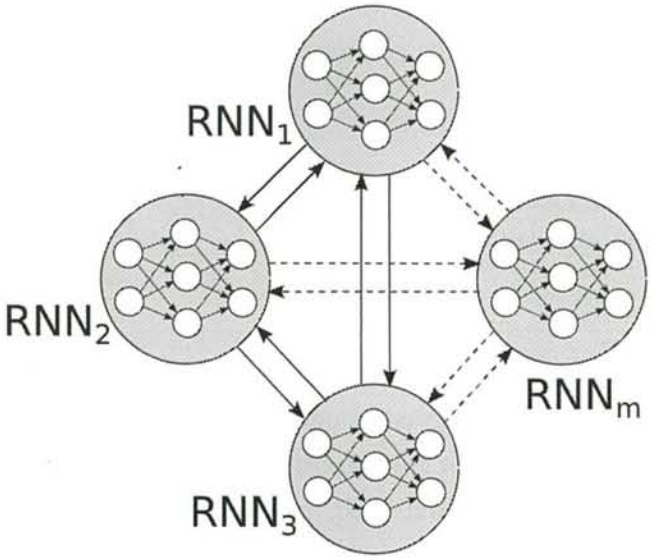



**Figure 6.16:** *Illustration model of MRNN*

### 6.3.4 MRNN model

In this section, new neuron based stabilization model is explained. In the previous research we used single model of learning system that assume in one certain condition represented all cases in the robot. Here, we used multimodal learning system that can assume different condition. We assume that robot will result different response in different condition. In the humanoid robot case, if the robot get a small disturbance such as the push or uneven terrain, then the robot only give hands response for protecting its stable condition. When robot get a high disturbance, then robot will response different action by positioning its footsteps. If the robot is often acquiring certain condition, then the robot has many experiences in that condition. Therefore, the robot has a good response when get experience in that certain condition.

Our MRNN model composed several RNN learning system. The learning system in this model work alternately depending on the condition of robot. The MRNN model can be seen in Fig. 6.16. There are $m$ RNN's composed the structure of MRNN which arrows represent the moving possibility between each RNN.

**Algorithm 6.1** Learning process in MRNN

```
Data input: tilt angle or angular velocity of the robot
Output: angular velocity of the motor
initialization
selection process
response action → delay time sampling
while not at end of the process do
  read current condition → evaluate previous state
  if Robot falling down then
    set the initial condition
  else
    selection process
    response action → delay time sampling
  end if
end while
```

The algorithm process of the proposed model is shown in the Algorithm 6.3. There are tilt angle or angular velocity of the robot as the input data, and the current speed acceleration of the robot wheels as the output data. First of all, the system select the current state of the RNN based on the current condition of the robot. Based on the current condition, the robot's action is activated and given delay time response. In the looping process, the robot read the current condition of the robot and evaluate the previous RNN based on the current condition. In this state, the system knows whether robot falling down or not. If the robot condition is in outside of the falling down condition then the system select the appropriate state of RNN. Based on the current condition, the system activate the response action, and time delay response is given. If the robot is falling down, then robot starts its initial condition.

The selection process is explained in Algorithm 6.4. Based on the current $\theta$ or $\dot{\theta}$ condition, system evaluate the weight parameters of $y_{t-1}$-th RNN. After that, the current id of RNN ($y_t$) is selected based on the Eq. 6.34. Where parameter $m_{rnn}$ is the number of RNNs. There are 2 method for placing the range of each RNN depicted in Fig. 6.18, first placing the desire angle point inside the RNN range. Second, placing the desire angle point in the RNN barrier.

$$y_t = int\left(\frac{(\theta + \theta_{min})}{\theta_{max} - \theta_{min}} m_{rnn}\right) \tag{6.25}$$

An example MRNN process can be seen in Fig. 6.17 where the desire point is 0.5. There are 5 RNN, each RNN has supported range where when the condition is inside the certain range of RNN then that RNN is activated. In this process illustration the first condition of the robot is 0.9 there for the selector state select RNN 1, after that RNN 1 send the output to

**Algorithm 6.2** Selection process

$\theta$ or $\dot{\theta}$ are the current condition of the robot
updating the weight parameters of $y_{t-1}$-th RNN in previous process
$y_t$ = the state id of RNN depending on the $\theta$ & $\dot{\theta}$
**if** selection parameter is $n$ **then**
  calculating process in $n$-th RNN
**end if**
response action

the robot. After 1 time sampling, the result condition is 8.0. this value is used for evaluation process of RNN 1 and the RNN selection of next sampling. Based on the illustration example, the running process flows from RNN $1 \rightarrow 2 \rightarrow 2 \rightarrow 4 \rightarrow 3 \rightarrow 4 \rightarrow 3 \rightarrow 3$.

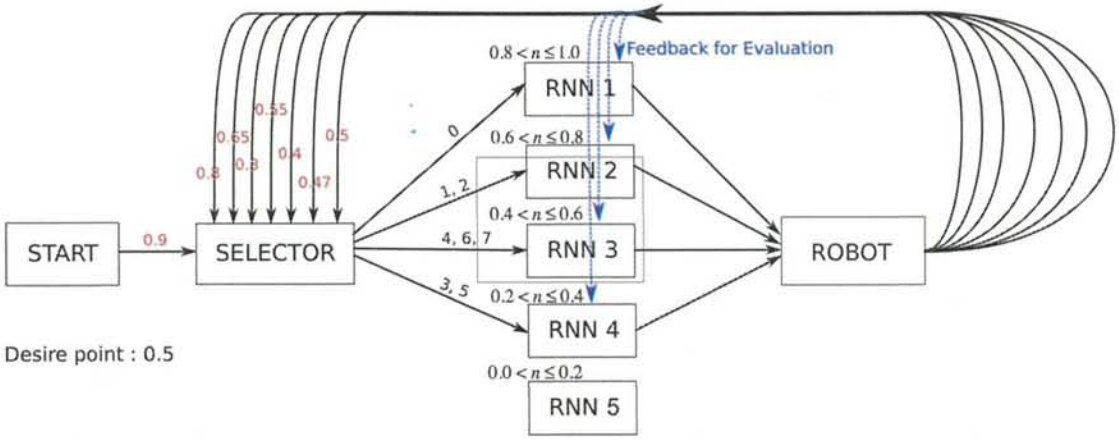



**Figure 6.17:** *Illustration of MRNN process*

### 6.3.5 Experimental Result

In these experimental results, we would proof the proposed stabilization model into 2-wheeled mobile robot. There are 2 manipulated output parameter in these parameter, which are acceleration or the speed as the output parameter of the wheels. In each manipulation, we compared the single RNN and MRNN with different number of RNN model.

In the first experiment we analyze the influence of number of RNNs toward the stability with setting the desire point not in the RNN barrier which its example can be depicted in Fig. 6.18b. In this experiment, we used 3 neurons in input layer, 20 neurons in hidden layer, 20 neuron in context layer, and 1 neuron in output layer. Where the 3 input neurons are required from previous neuron output ($O_k^y(t-1)$), Angular velocity of the robot's tilt ($\dot{\theta}$), and the previous wheel speed $v(t)$. The scenario of this experiment is, the proposed system should

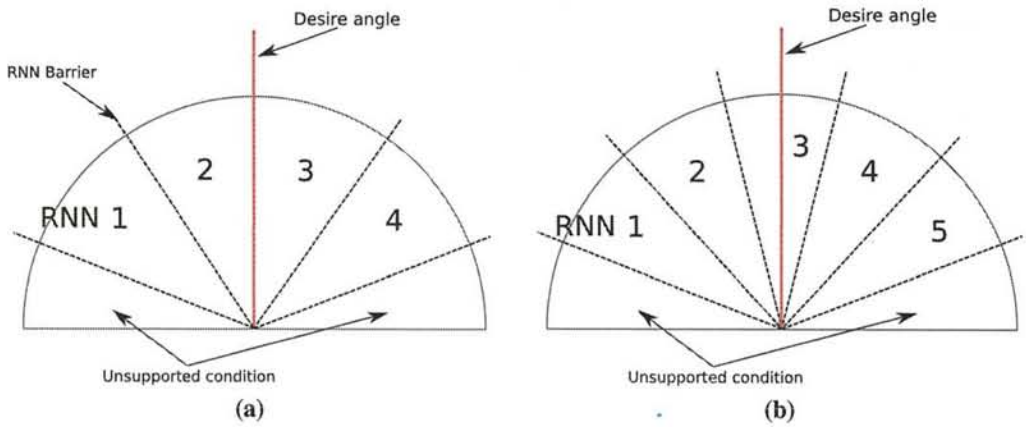



**Figure 6.18:** *The illustration RNN barrier a) desire angle in the RNN barrier b) desire angle inside the RNN range*

stabilize the robot and keep the desired tilt angle of the robot. We used Open Dynamics Engine in order to applied and analyze the proposed system in the simulation. We set the maximum number of time sampling as 10000.

We compare single RNN with MRNN with 5 RNNs, 11 RNNs, and 31 RNNs ($m_{rnn} = \{1, 5, 11, 31\}$). The input value and output value are normalized from 0 to 1. We set the learning rate $\eta$ as 0.5, $a_{max}$ and $a_{min}$ as $4\text{rad}/s^2$ and $4\text{rad}/s^2$, respectively, and $\theta_{max}$ and $\theta_{min}$ as $0.75$rad and $-0.75$rad, respectively. The result is depicted in Fig. 7.10. By using single RNN and MRNN with $m_{rnn}$ as 5, the stability point is not acquired until 10000 time sampling. By using 11 RNNs, the stability is acquired but the oscillation resulted is still high with oscillation amplitude is 0.4 radian. If we increase the number of RNN in MRNN system become 31, the amplitude oscillation decreased but the learning time took longer about 2000 time sampling, where MRNN with 11 RNNs took 500 time sampling. The number of RNNs effects the stability level of the robot application. In this experiment, big number of RNN indicates better stability level which has small oscillation amplitude.

Furthermore we tested the proposed stability model by putting desire point in the RNN barrier. We set the number of RNNs ($m_{rnn}$) as 2 and 4, and 40. The result is depicted in Fig. 7.12 showed better result than the previous experiment depicted in Fig. 7.10. In number of RNNs evaluation, there is no big different between small and big number of RNNs. Both of them are able to stabilize the proposed case in about 1000 time sampling. However, in order to learn multi desire angle, a large number of RNNs is required.

In order to prove the strength of stability system we tested the proposed system with certain number of RNNs by given the disturbance with certain power of push. In this experiment, we set the desire stability point in RNN barrier with 20 RNNs. After the robot reach

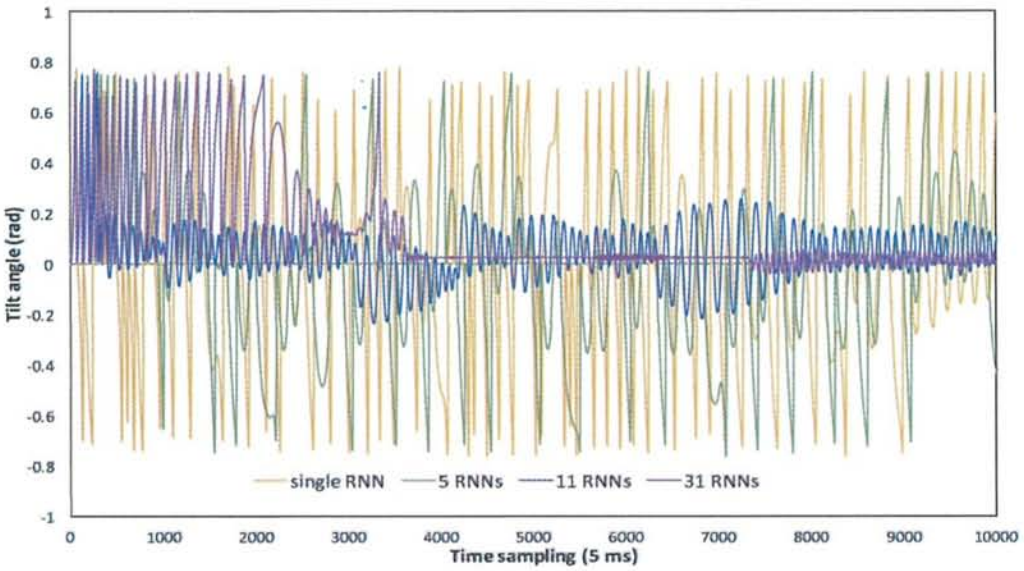



**Figure 6.19:** *The result of the learning process with different number of RNNs (desire point is inside RNN range)*

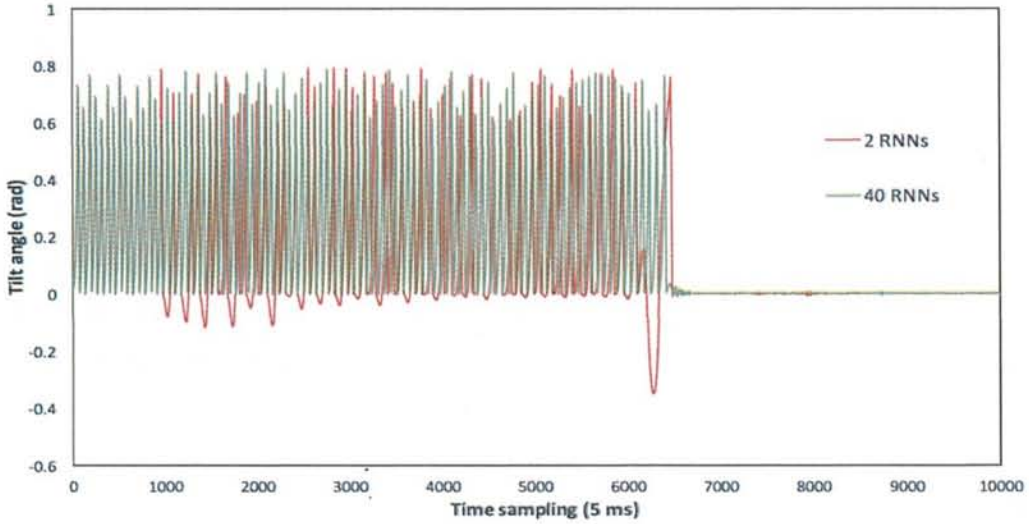



**Figure 6.20:** *The result of the learning process with different number of RNNs (desire point is in RNN barrier)*

the stable position without disturbance, the further teaching process was conducted by giving some disturbance until robot could stabilize from the disturbance. The result is depicted in Fig. 7.14 showed the oscillation of tilt body of robot in radian. After 15 disturbances were given or approximately 700 time sampling, the robot successfully stabilize toward the disturbance. When the robot got same disturbance like in $1600th$ and $2500th$ time sampling, the robot did not require to learn. It could stabilize because of the experience condition. If the robot frequently experience certain condition, the stabilization response is getting better. The sample of simulation of the proposed method are depicted in Fig. 6.22. Fig. 7.17a shows when the robot was falling down because got some disturbance and Fig. 7.17b shows the robot was trying stabilize the condition from disturbance.

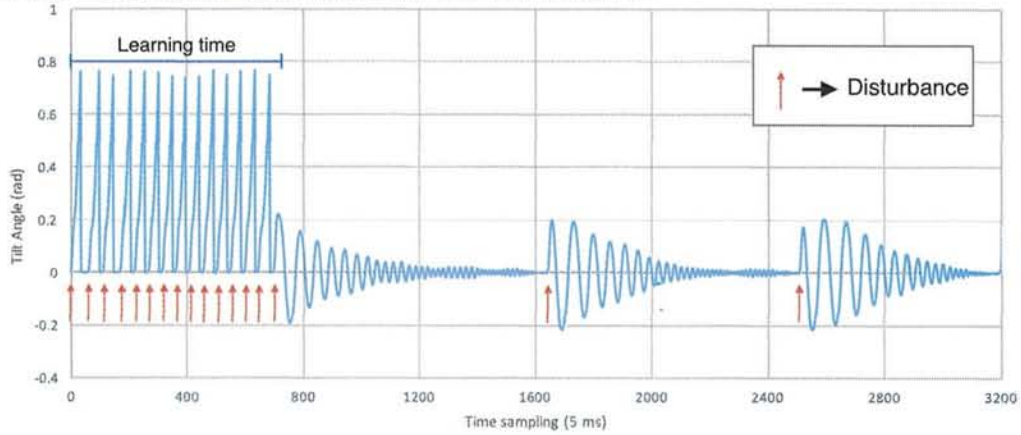



**Figure 6.21:** *The result of the learning process with given some disturbances*

Although this proposed system can stabilize in certain condition, this robot could not control its movement and the direction. However, in order to control the movement and control its direction, a lot of conditions should be trained. Therefore, after training process is done, the robot can control the movement by changing the desire point of robot's tilt angle. This proposed research is the basic of the balancing system of the biped robot locomotion. In the next development, this proposed research will applied into the humanoid robot with some modifying component, which are the speed of wheel will be changed by the speed of walking step. Therefore, by applied this proposed system into humanoid robot, it is expected can develop bio-inspired robust locomotion of humanoid robot.

### 6.3.6 Discussing

This research presents the bio-inspired stability system model. MRNN consist several RNNs where the number of RNNs is dependent on the complexity cases. There is selector system that decides certain RNN system depending the current condition of the robot. If the

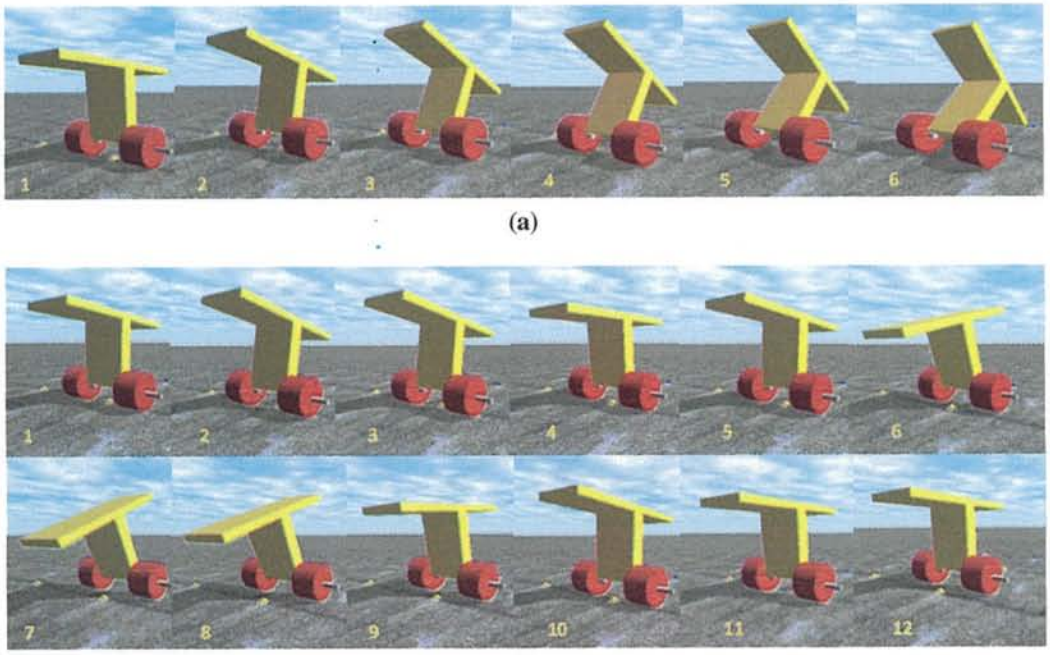



**Figure 6.22:** *Sample of simulations represent the experimental result in Fig. 7.10 with 11 RNNs (a) when robot was falling down (*$100 < time\ sampling < 150$*) (b) when robot was trying stabilize the disturbance (*$300 < time\ sampling < 400$*)*

robot often get the certain condition, then the RNN represent that certain condition has good respond for the robot. There are 2 methods for placing the range of each RNN, first, placing the desire angle point inside the RNN range and second, placing the desire angle point in the RNN barrier. There are influence of number of RNN in the first placing method. The large number of RNN indicates better stability level which has small oscillation amplitude. Single RNN was not enough to stabilize the proposed case. MRNN give better result than single RNN. In the second placing method there is no big different between small and big number of RNNs. However, big number of RNNs is required in order to learn multi-desire angle. The proposed model also successfully stabilized the robot from the disturbance in both placing methods. If the robot frequently experiences a certain condition, the stabilization response is getting better. Since the proposed stability model successfully applied in 2-wheeled robot, therefore this proposed model is able to be applied in biped robot stability.

## 6.4 Neuro-based Controller for Stability Behavior

This section presents neuro-based push recovery controller applied in humanoid biped robot in order to keep the stability with minimum energy required. There are three motion

patterns in human behavior when it gets external perturbation, those are ankle behavior, hip behavior, and step behavior. We propose the new model of modular recurrent neural network (MRNN) for performing online learning system in each motion behavior. MRNN consists of several recurrent neural networks (RNNs) working alternately depending on the condition. MRNN performs online learning process of each motion behavior controller independently. The aim of push recovery controller is to manage the motion behavior controller by minimizing the energy required for responding to the external perturbation. This controller selects the appropriate motion behavior and adjusts the gain that represent the influence of the motion behavior to certain push disturbance based on behavior graphs which is generated by adaptive regression spline. We applied the proposed controller to the humanoid robot that has small footprint in open dynamics engines (ODE). Experimental result shows the effectiveness of the push controller stabilizing the external perturbation with minimum energy required.

### 6.4.1 Introduction

Stability is the important part in bipedal humanoid robot. Robots have to walk stably in any condition and have to be ready getting external disturbance or internal disturbance. Following the humanoid robot development, the size of its footprint is getting smaller. It causes the walking of humanoid robot mainly not stable and its stability is difficult to be obtained. In the current state of the stability development, open loop based control is the most famous stability model applied in humanoid robot, especially in small humanoid robot. It results a walking model with good stability. However, pre-learning processes on well-defined surface are required. By using this technique, robot will be unstable when it finds undefined surface and undefined disturbance. Therefore, online stability learning model may be a great model in order to acquire a good stability [230].

In online based stability model, most researchers implemented physical method for the stabilization. They implemented inertial sensor or physical sensor and calculated the torques required in each joint for responding to any external disturbances. [82, 81] We also implemented physical approach in previous research. We applied inverted pendulum model and zero moment point in humanoid robot locomotion [176], [169]. On the other hand, researchers use biomechanical approach, applied the human behavior in order to recover and reach the stability level. Human shows the three different motion behaviors for responding sudden external disturbances, which are ankle recovery, hip recovery, and step recovery strategy [9, 186]. This algorithm is claimed that it has lower computational cost than physical based model because of its simplicity. Yi et al applied push recovery controller inte-

grated with three motion behaviors. However, these algorithms required a lot of training data [230, 231]. They also did not consider the energy required in stability activity that considered in this proposed research. Pre-defining the strategy classification is required to be performed before learning process, it is seemed as unnatural process in human behavior. [230] also has not been proved yet to be applied in humanoid robot that has human-like footprint. Our proposed model is therefore applied in human-like footprint or small footprint.

Furthermore, bio-inspired control system may be able to become a new innovation in push recovery controller. This algorithm shows the natural process of human model. Its effectiveness has been proved by several researchers [236], [62], [95]. Zhang et al shows the effectiveness of Recurrent neural network (RNN) as predictive control [236]. RNN was applied to control the stability of biped robot locomotion [115]. Neural network is also implemented in order to process sensory feedback information in central pattern generation (CPG) based locomotion [62]. Fukuda et al. combine RNN with evolutionary algorithm to achieve the stable locomotion in humanoid robot [59].

In previous research we successfully developed bio-inspired walking model in biped robot locomotion [168, 172]. It naturally follows human-walking mechanism, generates the angle of joints and its walking patterns are influenced by the environmental condition acquired from the sensory feedback. Therefore, in this proposed model, we applied bio-inspired stability controller for humanoid robot locomotion. This proposed stability model used multimodal learning system which can assume different condition. It assumes that robot performs different behavior in different condition. In the humanoid robot case, if the robot gets a small disturbance such as the push or uneven terrain, then the robot only gives hands response for protecting its stable condition. Three motion behaviors in biomechanical approach is also considered in this proposed model.

The contribution of our proposed model is applying natural process of the human learning process by using neuro-inspired stability controller. Self-adaptation process is implemented with few pre-defined parameters. Proposed controller is consider the energy required in performing stability activity where most researcher did not consider energy minimization. This controller is able to choose appropriate motion behavior in order acquired minimum energy required. Another superiority of the proposed controller is its capability to be applied into small footprint humanoid robot.

### 6.4.2 Motion Behavior Controller

The proposed push recovery controller has three motion respond behaviors. Each motion behavior implements modular recurrent neural network (MRNN) as the online learning

controller. Each MRNN results parameters represent the certain motion behavior. During the learning process of motion behaviors, each MRNN performs independently. Therefore we acquire the best response of each MRNN in certain condition or when acquiring the disturbance. After that, push recovery controller gives the portion or each MRNN to respond to the certain condition.

#### 6.4.2.1 RNN

In this section, we explain how MRNN's system is implemented for two-wheeled mobile robot. Many researchers have used RNN model for control system. In the previous research, we used single RNN in order to build the stability system in humanoid robot [168]. In this proposed model, MRNN is composed of more than one RNN system which works alternately depending on the certain condition in the certain time. RNNs process their previous condition in current process, therefore the previous condition will influence and become input in the feed-forward process.

In this research, the RNN system is divided into four layers which are input layer, hidden layer, context layer, and output layer. The number of hidden and context layers are same since context layer represents the previous condition of hidden layer. The mathematical model of feed forward process of RNN can be seen in Eq. (6.27) and (6.28). In Eq. (6.27) and (6.28), $X_i(t)$ is the input value in certain time from the sensor feedback from the $i$th input neuron, $S'_j(t)$ and $S_j(t)$ are the inputs and outputs of the $j$th hidden neuron, and $O'_k(t)$ and $O_k(t)$ are the inputs and outputs of the $k$th output neuron. The number of neurons in the input layer, in the hidden layer, and in the output layer are denoted by $m$, $p$, and $n$, respectively. Parameter $y$ shows the current RNN activated.

$$e = \begin{cases} 0 & \text{if } \alpha_t < (\alpha_{t-1} - \beta) \\ \text{sgn}\big(d_k - g(O_k(t))\big)\big(\alpha_t - (\alpha_t - \beta)\big) & \text{otherwise} \end{cases} \tag{6.26}$$

$$S_j^y(t) = f(S'_j(t)) = f\left(\sum_i^m X_i(t) A_{ij}^y(t) + \sum_h^p S_h(t-1) B_{hj}^y(t)\right) \tag{6.27}$$

$$O_k^y(t) = f(O'_k(t)) = f\left(\sum_j^p S_j(t) C_{jk}^y(t)\right) \tag{6.28}$$

In this case, the desired output is the condition where the robot maintains the angular velocity at zero value. $\delta_k$ is the error propagation in the output node and $\delta_j$ is the error propagation in the hidden node computed in Eqs. (6.29) and (6.30), $e$ is the error value

calculated in Eq. 6.26, where $\alpha = \text{abs}(d_k - g(O_k(t)))$, $d_k$ is the desired output, and the output function $g(x)$ is the output of the sensor.

$$\delta_k^y = (e_t) f'(O_k'(t)) \tag{6.29}$$

$$\delta_j^y = \sum_k^n \delta_k^y C_{jk}^y(t) f'\left(S_j'^y(t)\right) \tag{6.30}$$

$$\mathbf{C}^y(t+1) = \mathbf{C}^y(t) + \eta \mathbf{S}^y(t) \delta_k^y \tag{6.31}$$

$$\mathbf{B}^y(t+1) = \mathbf{B}^y(t) + \eta \mathbf{X}^y(t) \delta_j^y \tag{6.32}$$

$$\mathbf{A}^y(t+1) = \mathbf{A}^y(t) + \eta \mathbf{S}^y(t-1) \delta_j^y \tag{6.33}$$

In Eq. (6.27), the activation function for the hidden layer, $f(x)$ uses sigmoid function. In Equations (7.10), (6.32), and (7.2), the weight parameters of the neurons are presented by **A**, **B**, and **C** matrices acquired by a learning process. In BPTT, the error propagation is done recursively, where $\eta$ is the learning rate of the weight of synapse between the motoric and the sensoric neuron. Those weight parameters can be dynamically regenerated depending on the environmental condition.

In this RNN model, 2 neurons are used in input layer as the tilt or angular velocity angle and the previous speed acceleration of the wheels. Hidden layer and context layer consist 10 neurons, while in output layer consist 1 neuron represents the speed acceleration of the robot.

#### 6.4.2.2 MRNN model

In this subsection, we explain the learning model in order to optimize the motion response behavior. The different condition can be considered by using the proposed MRNN. Robot will perform different responses in different conditions. If the robot is often acquiring certain condition, then the robot has many experiences in that condition. Therefore, the robot has a good response when get experience in that certain condition. In the learning process, robot will be given various conditions to get good diversity of experiences.

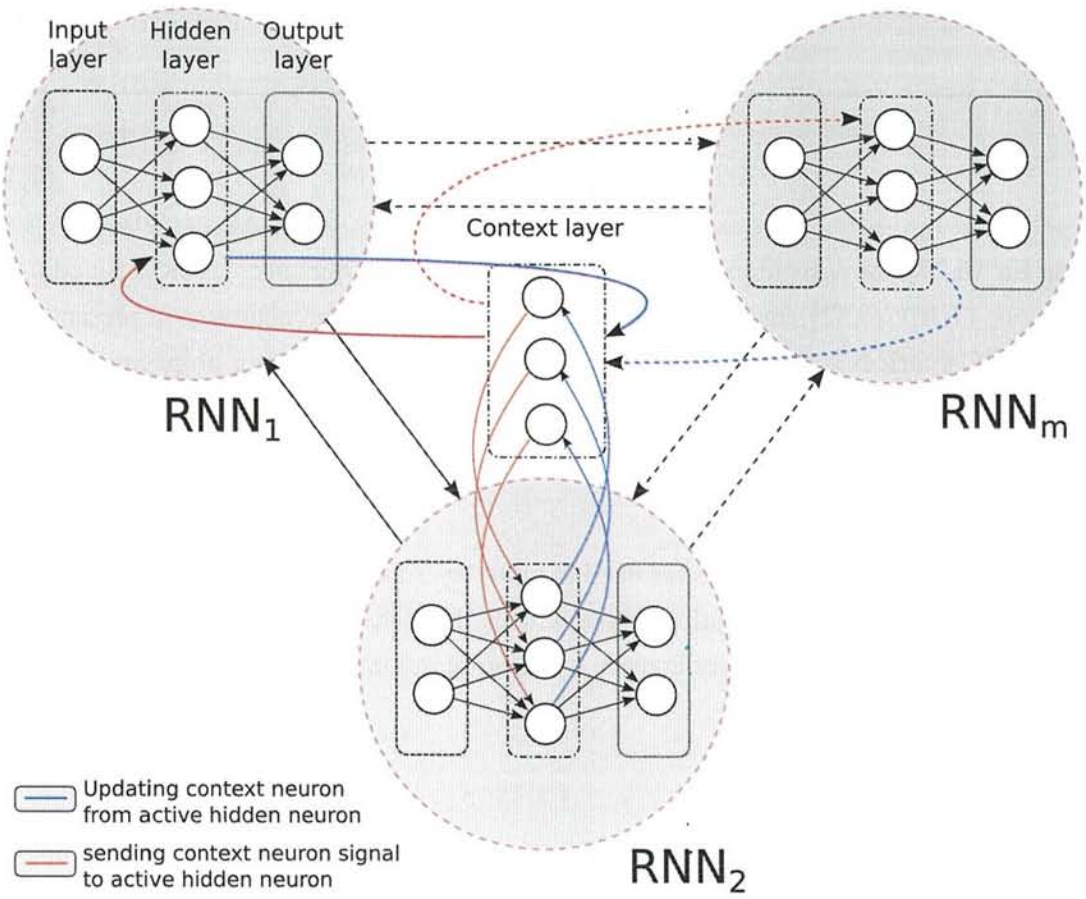



**Figure 6.23:** *Design of MRNN model, RNNs have one sharing context layer*

**Algorithm 6.3** Learning process in MRNN

```
Data Input : tilt angle and CoG of the robot
Result : motion behavior
initialization
selection process
response action → delay time sampling
while not at end of the process do
  read current condition → evaluate previous state
  if Robot falling down then
    set the initial condition
  else
    selection process
    response action → delay time sampling
  end if
end while
```

This MRNN model is composed of several RNN which work alternately depending on the condition of robot. In this MRNN model, RNN has independent input, hidden, and output layer, but they has same context layer. Context layer provides the value of previous active hidden layer. The MRNN model can be seen in Fig. 6.23. In Fig. 6.23, blue line presents the updating process of neuron in context layer from previous active neuron in hidden layer and red line presents the sending process to active neuron in hidden layer. The structure of MRNN is composed of $m$ RNN which arrows represent the moving possibility between each RNN.

The algorithm process of the proposed model is shown in the Algorithm 6.3. There are tilt angle or angular velocity of the robot as the input data, and the current speed acceleration of the robot wheels as the output data. First of all, the system selects the current state of the RNN based on the current condition of the robot. Based on the current condition, the robot's action is activated and given delay time response. In the looping process, the robot reads the current condition and evaluate the previous RNN based on the current condition. In this state, the system knows whether robot is falling down or not. If the robot condition is outside of the falling down condition then the system selects the appropriate state of RNN. Based on the current condition, the system activates the response action, and time delay response is given. If the robot is falling down, then robot starts its initial condition.

$$y_t = \text{int}\left(\frac{(\theta + \theta_{min})}{\theta_{max} - \theta_{min}} m_{rnn}\right) \tag{6.34}$$

The selection process is explained in Algorithm 6.4. Based on the current $\theta$ condition, system evaluates the weight parameters of $y_{t-1}$-th RNN. After that, the current id of RNN ($y_t$) is selected based on the Eq. 6.34. Where parameter $m_{rnn}$ is the number of RNNs. There are 2 methods for placing the range of each RNN depicted in Fig. 6.18, the first is placing

the desire angle point inside the RNN range and the second is placing the desire angle point in the RNN barrier.

**Algorithm 6.4** Selection process procedure

1: $\theta$ or CoG are the current condition of the robot
2: updating the weight parameters of $y_{t-1}$-th RNN in previous process
3: $y_t$ = the state id of RNN depending on the $\theta$
4: **if** selection parameter is $n$ **then**
5: calculating process in $n$-th RNN
6: **end if**
7: response action

#### 6.4.2.3 Motion behaviors

Based on the preliminary studies, each motion behaviors have different considered parameters. Therefore, MRNN of motion behavior has different parameters to be optimized and also has different input parameters represent the condition. The output values from MRNN are normalized from 0 to 1. However there are different range output data in each behavior strategy. Parameter $a(i)$ normalize the output value in $i$th output data computed in Eq. (6.35), where constant parameters $a^{(max)}$ and $a^{(min)}$ are tabulated in Table 6.6. We conducted online learning process of each motion behavior independently because the MRNN system only provides one considered parameter as the output. Multiple considered parameters will become our future research. In this motion behavior, we minimize the stability time required in recovering process. We calculate the torque values in each joint during recovering process.

$$a(i) = (a_i^{(max)} - a_i^{(min)}) + a_i^{(min)} \tag{6.35}$$

**Table 6.6:** *Constant parameter of maximum and minimum value*

| $i$ | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| $a_i^{(min)}$ | $-\pi/9$ | $-\pi/3$ | $-\pi/6$ | $-\pi/4$ | $-\pi/2$ | $-90$ | 0 |
| $a_i^{(max)}$ | $\pi/9$ | $\pi/3$ | $\pi/6$ | $-\pi/4$ | $-\pi/2$ | 90 | 1.0 |

**6.4.2.3.1 Ankle behavior** In ankle behavior, robot uses its ankle to respond to the disturbance. The angles of ankle joint are controlled to keep the center of gravity (CoG) inside the supported area. Hand joint is also controlled for supporting the ankle joint. Ankle behavior is represented by $\text{MRNN}_1$, where it considers walking speed and CoG point ($p^{CoG}$) as the input parameter and parameter $x^{(ankle)}$ in joint angle level as the output parameter. Therefore

ankle joint ($\theta^{(ankle)}$) is calculated in Eqs. (6.36) and (6.36). Where $\Theta^{(ankle)}$ represents the joint angle generated by locomotion generator, $G_1$ represents gain value in ankle controller. Since the CoG is the controlled parameter, desire value $d_k$ in RNN model is minimization of the CoG point deviation.

$$\theta^{(ankle)} = \Theta^{(ankle)} + G_1 \cdot x^{(MRNN_1)} \cdot a(1) \tag{6.36}$$

$$\theta^{(hand)} = \Theta^{(hand)} + G_1 \cdot \alpha_3 \cdot x^{(MRNN_1)} \cdot a(2) \tag{6.37}$$

In this strategy, one RNN has two neurons in input layer, eight neurons in hidden layer, and one neuron in output neurons. The number of neuron in context neuron is same as the number of neuron in hidden layer.

**6.4.2.3.2 Hip behavior** Hip controller controls knee and hip joint and combines with ankle behavior. Hip behavior also occupies hand joint to support the robot to respond to the disturbance. Those joint angles are also controlled to keep the CoG point inside the supported area. Hip behavior is represented by $\text{MRNN}_2$ which also considers CoG point ($p^{CoG}$) as the control and input parameter. The calculation of joint angle represents the hip behavior calculated in Eqs. (6.38), (6.39), and (6.40), where $\Theta$ is resulted from locomotion generator, $G_2$ represents gain value in hip controller. Based on the preliminary experiments, we set the constant parameters ($\alpha_1$, $\alpha_2$, $\alpha_3$) for dividing the portion of the angles. In this MRNN, each RNN has two, ten, and one neuron in input layer, hidden layer, and output layer respectively.

$$\theta^{(hip)} = \Theta^{(hip)} + G_2 \cdot \alpha_1 \cdot x^{(MRNN_2)} \cdot a(3) \tag{6.38}$$

$$\theta^{(knee)} = \Theta^{(knee)} + G_2 \cdot \alpha_2 \cdot x^{(MRNN_2)} \cdot a(4) \tag{6.39}$$

$$\theta^{(hand)} = \Theta^{(hand)} + G_2 \cdot \alpha_3 \cdot x^{(MRNN_2)} \cdot a(5) \tag{6.40}$$

**6.4.2.3.3 Step behavior** Step behavior controls the step position ($P$) and stepping time in locomotion generator that was explained in our previous research [169]. Those parameters are controlled based on the input parameters, the CoG condition and the walking speed. This behavior strategy is represented by MRNN as the controller model, where each RNN is structured as follows: 2 neurons input layer; 10 neurons in hidden layer; one neuron in output layer.

$$S_f = G_3 \cdot x^{(MRNN_3)} \cdot a(6) \tag{6.41}$$

$$t^{(walk)} = t^{(locomtion)} + \text{abs}(x^{(MRNN_3)} \cdot a(7)) \tag{6.42}$$

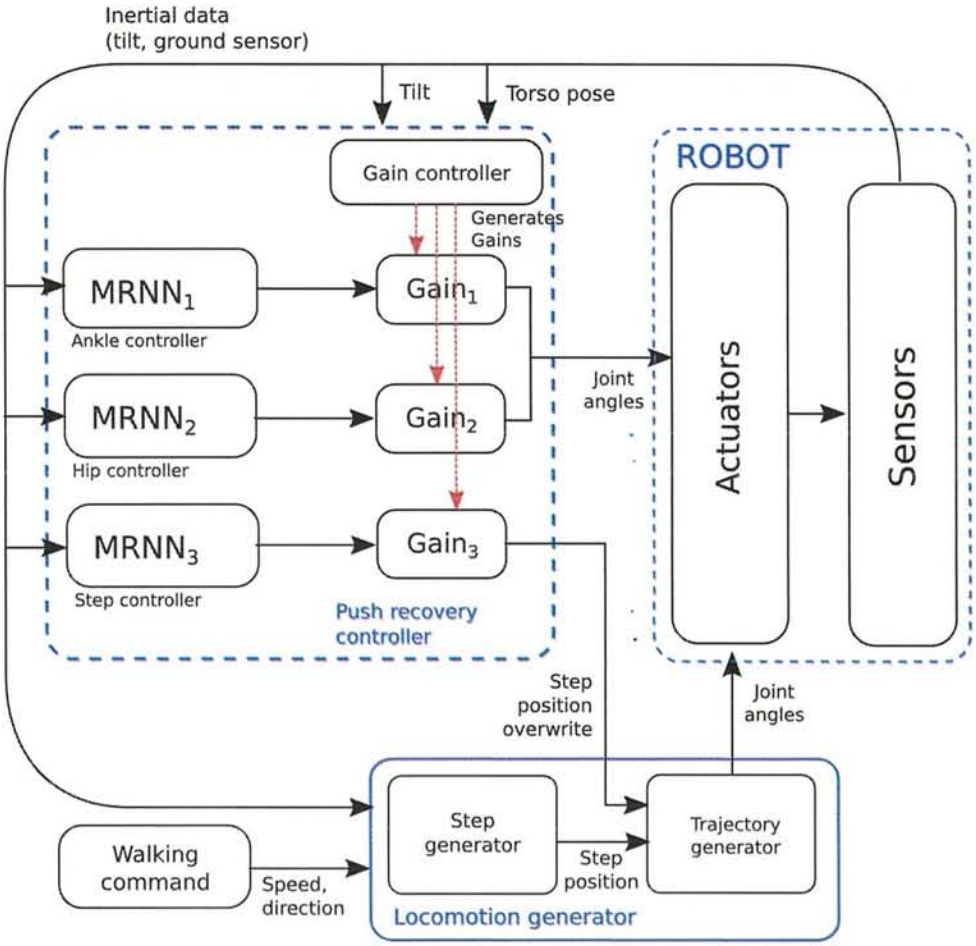



**Figure 6.24:** *Model of push recovery controller*

If there is high disturbance, then the step behavior controller overwrites the step length and increases the incremental time in locomotion generator system. Step length ($S_f$) and incremental time ($t^{(step)}$) are computed in Eqs. (7.8) and (6.42), where $t^{(walk)}$ is the incremental time of step, $t^{(locomtion)}$ is incremental time generated by locomotion generator, and $G_3$ represents gain value in step controller.

### 6.4.3 Push recovery controller

Push recovery controller manages and chooses the appropriate motion behavior depending on robot condition. The design of the push recovery controller can be seen in Fig 6.24. This controller adjusts the gain that represents the influence of the motion behavior to certain disturbance. The aim of this controller is to manage the motion behavior controller by minimizing the energy required for responding the external perturbation.

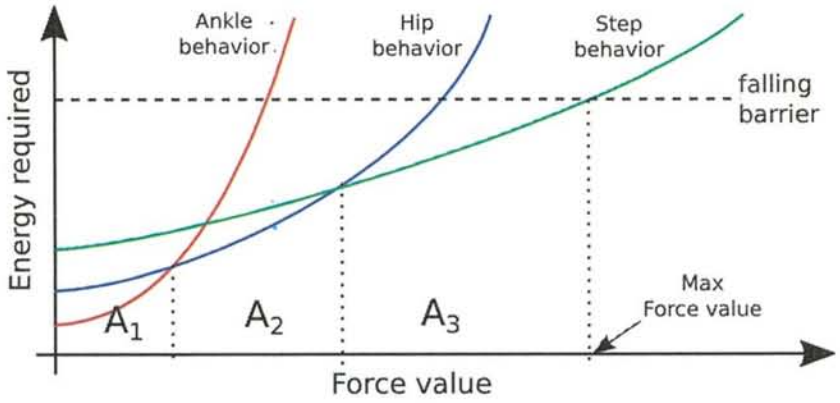



**Figure 6.25:** *The prediction of graphics relationship between force input and energy required*

In order to acquire initial data of appropriate motion behavior, the system conducts several preliminary training by giving the push with random force value in each motion behavior and calculates their required energy. Furthermore, the energy required in each motion behavior will be online updated. Push recovery controller will adjust the appropriate motion behavior by using regression analysis model. Multivariate adaptive regression splines (MARS) is the best regression model for this proposed model [72, 55]. This model results the graphics relationship between the force value of push and the energy required in each motion behavior, where its sample graphic can be seen in Fig. 6.25. The pattern of graphics will adjust along with updating the motion behavior data. In Fig. 6.25, ankle motion behavior (AMB) is active when force condition in A1 range force value, hip motion behavior (HMB) is active when force condition in A2 range force value, and step motion behavior (SMB) is active when force condition in A3 range force value.

$$f_{(i)}(X) = \beta_0 + \sum_{m=1}^{M} \beta_m \prod_{k=1}^{K_m} \left[ i_{(k,m)} (x_{(k,m)} - t_{(k,m)}) \right]_+^q \tag{6.43}$$

$$\left[ -(x_k - t_k) \right]_+^q = \begin{cases} ll(t_k - x)^q & : x < t_k \\ 0 & : x \geq t_k \end{cases} \tag{6.44}$$

$$\left[ +(x_k - t_k) \right]_+^q = \begin{cases} ll0 & : x < t_k \\ (x - t_k)^q & : x \geq t_k \end{cases} \tag{6.45}$$

The push recovery controller will adjust the gain parameter ($G_i$) by calculating minimum predictive energy required by each motion behavior in certain disturbance, which is calculated in Eq. (6.43), (6.44), (6.45). The detail explanation regarding the MARS model can be seen in [72] and [55]. The smallest value of $f_{(i)}(X)$ represents the selected motion behavior,

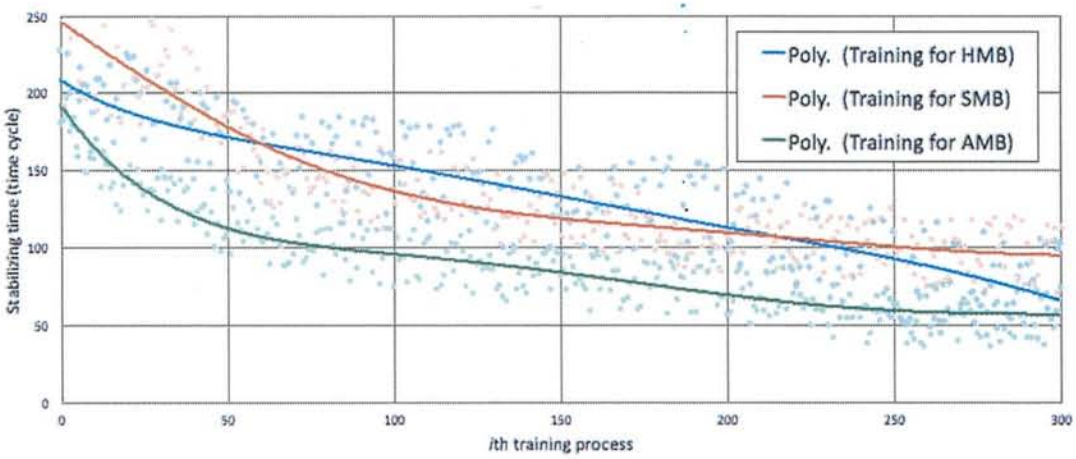



**Figure 6.26:** *Training process of MRNN*

where ($i = 1$, 2, and 3) represent the AMB, HMB, and SMB, respectively.

In the learning process of push recovery controller, initial training data is required. The robot is trained by giving the force disturbance with certain motion behavior. We calculate force value and energy required for stabilizing the robot. This initial data are became the first reference of the MARS predictive graph that is used for deciding the motion behavior if get a disturbance with certain force value. After that, robot will get various force disturbance during the robot activities. The graph will be updated and adjusted therefore result better and appropriate motion behavior when get certain disturbance.

### 6.4.4 Experimental result

Experimental result shows the several experiment regarding push recovery model. First of all, we conducted independently the learning process of MRNNs represent motion behaviors. In this experiment, we set time cycle as 0.05 second and learning rate in RNN ($\eta$) as 0.01. In this experiment, we minimize the error and required stability time shown in Fig. 6.26. MRNN model succeeded minimizing the error and time required of stability when getting the disturbance. Fig. 6.26 shows the decreasing of required stability time after getting the disturbance. High required stability time indicates robot was not stable and tended to falling down. The result of robot performance in different motion behavior when get force disturbance can be seen in Fig. 6.27.

After all motion behaviors have been trained and optimized, we conducted optimization process for push recovery controller. In order to acquire initial data for initial graph generated by MARS, the robot is trained by giving the force disturbance with certain motion behavior.

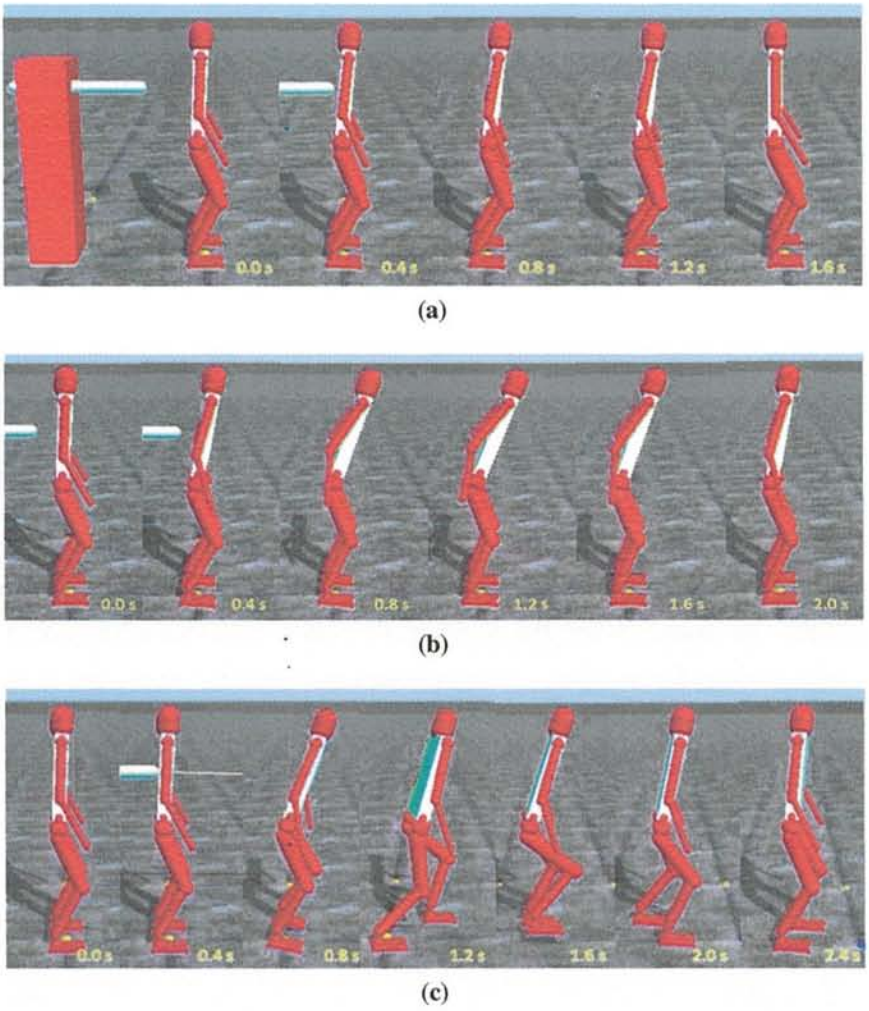



**Figure 6.27:** *Robot performed the stability respond (a) ankle motion behavior (b) hip motion behavior (c) step motion behavior*

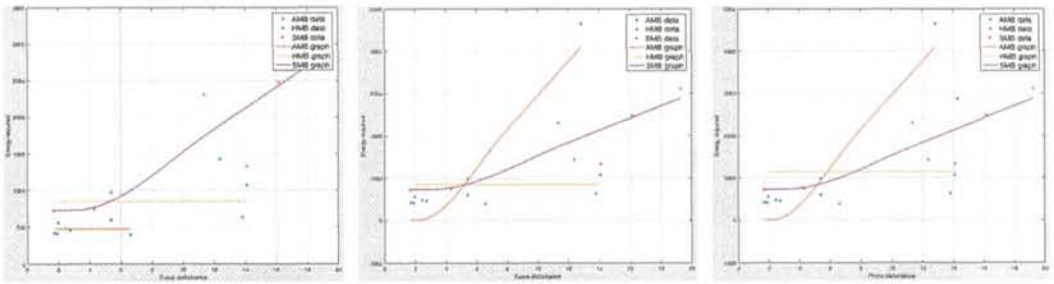
In this experiment, each motion behavior controller (Ankle, Hip, Knee controller) was given six push disturbances with various force value. Therefore, there were 18 training data are performed. After that, first graph reference in each behavior was generated shown in Fig. 6.28a. This graph is for deciding the motion behavior activated for the next force disturbance. In this stage of push recovery controller development, gain values ($G_{i\ (i\in 1,2,3)}$) are set as binary output calculated by step function ($S(x)$) in Eq. (6.46).

$$\begin{aligned} G_1^{(t)} &= S\big(\min\big[\delta[2,1],\delta[3,1]\big]\big) \\ G_2^{(t)} &= S\big(\min\big[\delta[1,2],\delta[3,2]\big]\big) \\ G_3^{(t)} &= S\big(\min\big[\delta[1,3],\delta[2,3]\big]\big) \\ &\text{Where :} \\ \delta[i,j] &= (f_{(i)}^{(t-1)}(F(t)) - f_{(j)}^{(t-1)}(F(t))) \end{aligned} \tag{6.46}$$

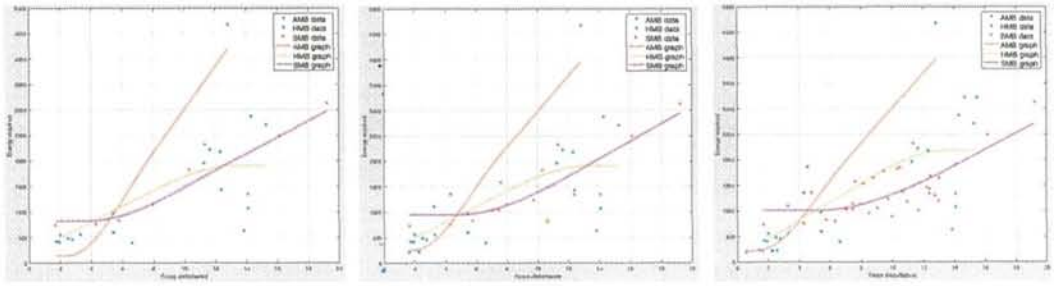
In Eq. (6.46), $F(t)$ is the force value of disturbance in current time, $f_{(i)}^{(t-1)}$ is the predictive regression graph generated by MARS based on the previous data. Graph predictive of certain motion behavior will be updated after motion behavior is activated based on the gain value. During robot activities, robot was given many disturbance with various force value. The graph of behaviors were updated and adjusted that depicted in Fig. 6.28. Therefore push recovery controller generated better and appropriate motion behavior after getting a hundred experiences with various force value. After performing a hundred experiences the structure of behavior graphs was not changed, because the graphs have been well optimized. The equation result of graph after a hundred experiences can be seen in Eqs. (6.47).

$$\begin{aligned} f_{(\mathrm{AMB})}(x) &= 311.72 + 421.32 \cdot \big[(x-1.4208)\big]_+ \cdot \big[(x-2.2601)\big]_+ \cdot \big[(x-6.5033)\big]_+ \\ f_{(\mathrm{HMB})}(x) &= 659.55 + 203.221 \cdot \big[(x-3.1851)\big]_+ \cdot \big[(x-5.5455)\big]_+ \cdot \big[(x-10.515)\big]_+ \\ f_{(\mathrm{SMB})}(x) &= 1167.7 + 90.443 \cdot \big[(x-5.0513)\big]_+ \cdot \big[(x-8.4261)\big]_+ \cdot \big[(x-13.8285)\big]_+ \end{aligned} \tag{6.47}$$

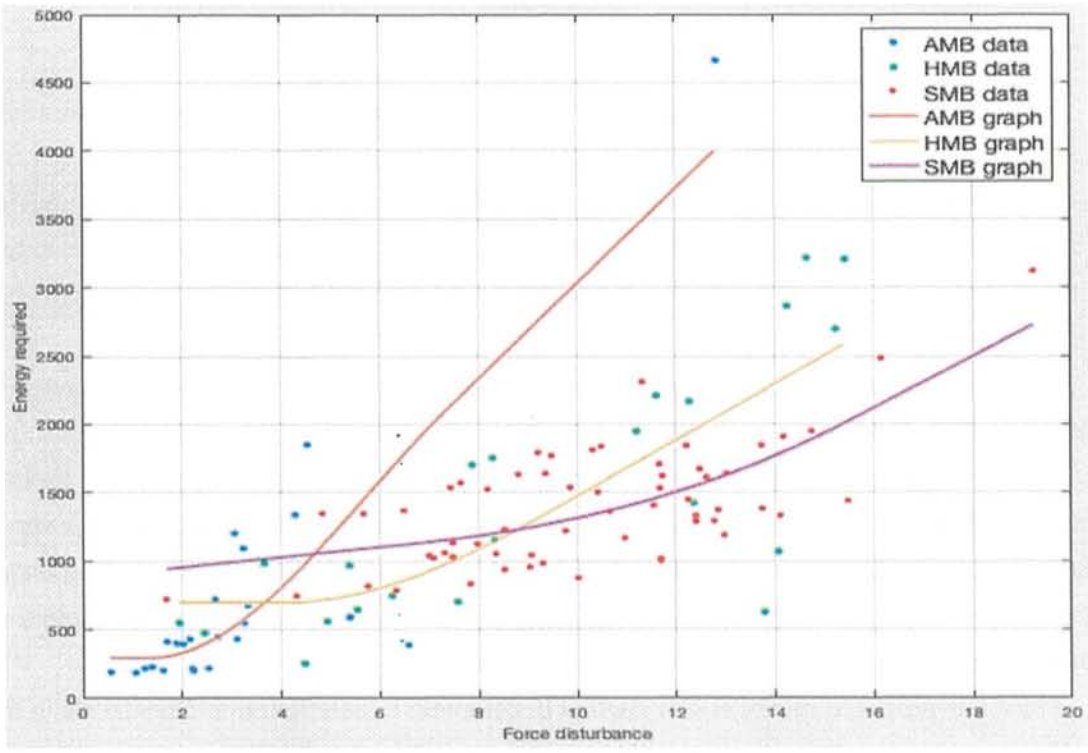
Push recovery under external perturbation was also tested to the robot during performing the movement. Fig. 6.29 shows the good response performed by push recovery controller when get a disturbance with high force value. Push recovery controller is effectively applied in humanoid robot as the online stability learning system (SLS). Compared with current push recovery model [230] [82, 186], this controller can online optimize the appropriate behavior based on the value force of disturbance.



(a) (b) (c)



(d) (e) (f)




(g)

**Figure 6.28:** *(a) Initial behavior graph with initial reference data (b) the changing of behavior graph with 20 data reference (c) 25 data reference (d) 30 data reference (e) 50 data reference (f) 70 data reference (g) optimized graph behavior is formed in a hundred data reference*

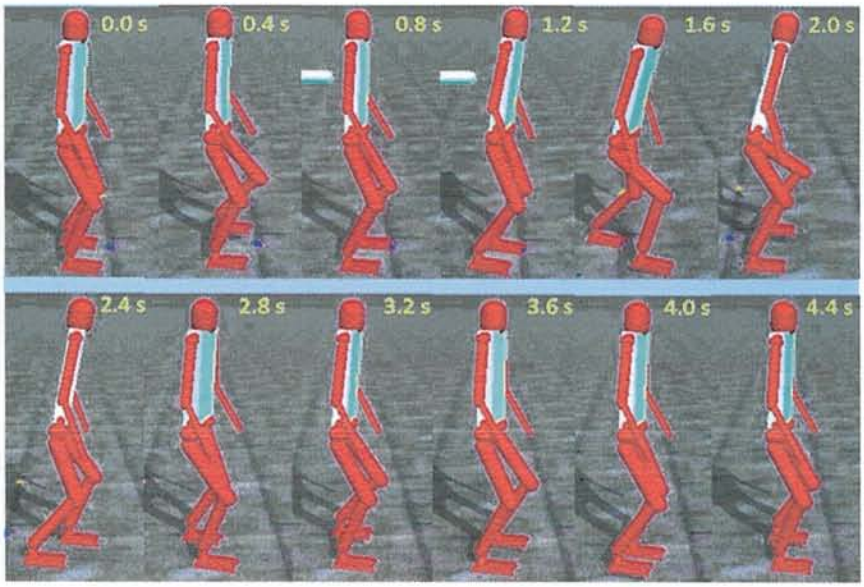



**Figure 6.29:** *Push recovery testing under external perturbation when walking at speed of 50 cm/second*

### 6.4.5 Conclusion

This research proposed push recovery model in humanoid robot based on neuro-activity model. MRNN that contains several RNNs are proposed for stability learning systems (SLSs) in three motion response behaviors, which are ankle, hip, and step behavior. One behavior is represented by one MRNN. MRNN successfully creates stabilized behavior under external disturbance. However further research is required to increase the number of considered output value.

Push recovery controller is proposed for managing an appropriate motion behavior in order to respond to certain push disturbance. Online learning algorithm applied in push recovery controller based on MARS model successfully forms and decides appropriate motion behavior after a hundred given experience perturbation with random force value. Experimental result also shows the effectiveness of the model stabilizing the robot walking under external perturbation. Based on that, push recovery controller is effective to be applied in humanoid robot as the online stability learning system.

Since this proposed model is still applied in computer simulation, implementation in the real humanoid robot becomes a possible homework for future research.

# Chapter 7

# Neural Based Path Planning

In this chapter, we wiil show a new motion planning model that inspired by neural activity. There are two transmission process of neural model, forward transmission and backward transmission process. In the experimental result, integration of the motion capabilities models will be applied into simulated and real four legged robot.

## 7.1 Introduction

During a disaster, the terrain, route, area, and also building are becoming unstructured, diverse, and challenging. It makes the rescuing process difficult and challenging for human or robot who is in charge in disaster area. The route in disaster area becomes unstructured, making it difficult for the rescuer whether human or robot to find the best and possible route in the rough and unstructured terrain. Since it is very dangerous for human to rescue in unstable environment, robot is the best choice for exploring and investigating the disaster area. Most researchers proposed wheeled mobile robot for rescuing in disaster areas [139, 159, 182], but legged robot is more effective in rough terrain [11]. Therefore, we applied this proposed algorithm in a four-legged robot which is equipped with several supporting sensors. Four-legged robots are more efficient than biped robots in the stabilization cases and are able to achieve the maximum movement speed compared to robots with more legs. Furthermore, 3-D path planning model is required in order to support and facilitate those robots while moving on rough terrain. Therefore, in this research, we propose an online 3-D path planning optimization based on neural activity.

Three dimensional path planning studies have used images for path planning and also applied multiclass support vector machines for obstacle avoidance [137]. By using this idea, the robot can generate the safe pathway. In [179], 3-D based path planning was proposed,

and D* algorithm was modified in order to estimate the distance in sloping terrain. Beside that, Dogru et al proposed genetic algorithm for optimizing pathway with minimum energy required [43, 44]. However, [179, 43, 44] did not consider the unpredictable obstacle. Beside that, Kroumov et al solved the obstacle problem but there is still problem with concave 3-D obstacles [103]. 3-D path planning was also proposed for UAV movement algorithm [145, 166, 211]. These algorithms were applied in computer simulation. In the integration system, UAV or drone is used for generating the map. Some researchers used 3-D map reconstruction in order to acquire the 3-D map model and to find the best pathway based on the result of reconstruction map. The robot was equipped with laser range finder (LRF) or Kinect sensor in order to support the reconstruction algorithm. [26] also used LRF to generate the 2-D maps. This idea deals with static environment. Henry et al used Kinect camera in order to model 3-D indoor environment [74]. In the previous research, multi-resolution map was also used for decreasing the computational cost existing in high resolution map of path planning [201]. After that, a real-time feature extraction and segmentation method for a 3-D map [202] was proposed in order to increase the efficiency of the topological map. This map model can decrease the memory usage for reconstructing the map. Therefore, [202] is suitable to be combined for the proposed 3-D path planning model in the next stage.

In further cases of the environment model generated by LRF and Kinect sensor, the unpredictable surface such as friction, softness of the terrain, and unpredictable collision sometimes become the problem. Those sensors are used as initial data in path planning, therefore those ones only deal with static path planning. In the real cases, the robot deals with dynamic environment, and it should autonomously work through dynamic environment. When the algorithm deals only with static environment, the robot will get problem when it finds an unpredictable collision. Therefore, the dynamic path planning is important in order to deal with dynamic environment. Dynamic path planning will regenerate the path planning when the algorithm finds unpredictable condition or changing environmental condition. In order to support the dynamic path planning, either additional sensors in mobile robot or measurement tools for human are required to measure and detect unpredictable collisions (friction, obstacle, and softness of terrain) which cannot be considered in initial map generated by LRF or Kinect sensor.

Most mobile robots (wheeled and legged) are supplemented with the capability for finding the pathway. Some people used traditional algorithm such as: A*, D*, and Dijkstra algorithm in order to find the best pathway [114], [53], [143], [42], [113]. Ferguson et al proposed a modified D* algorithm in order to reduce the path cost in non-uniform environment. An interpolation equation was proposed in order to cut the inefficient pathway [53]. However, in [53]'s algorithm more computational cost was required. In common cases, D* path planning

algorithm is faster than A*, but [75] modified the adaptive A*, therefore it could be faster than D* in some cases. These algorithms [75, 53] required predefined travel cost, therefore these algorithms seem difficult to cover unpredictable obstacles.

However, the traditional algorithm of path planning requires the determination of the path planning rule and it uses a recursive algorithm. Therefore, these algorithms require high computational cost. Bio-inspired algorithm provides a natural process that is able to dynamically generate the best pathway [160, 154]. Some researchers have proved the effectiveness of neural based model for path planning problems [64], [224], [227], [154]. In [64], Hopfield method was applied in a neural network to generate the pathway with obstacle avoidance. This proposed model was applied in a 2-dimensional simulation. Neural based approach can also be applied for complete coverage path planning with obstacle avoidance [224]. Quoy et al proposed the control model in mobile robot by using dynamical neural networks based on the neural field formalism that was applied in the mobile robot path planner [154]. Most of the researchers applied pulsed neural network in order to find the efficient pathway [152], [153], [238, 222]. They focused on the inner activity of the neuron. Qu et al proposed pulsed neural network to generate real time collision free path planning [152]. Zhang et al applied a simple pulse coupled neural network to decrease the computational cost, but the defined travel cost is required in this case [238].

Furthermore, modified pulsed neural networks were proposed by several researchers in order to increase the performance of the path planning algorithm [111, 67, 153]. In [67], quick optimization process of path planning was proposed by using PCNN model. However, there is no efficient model to reduce the number of evolved neurons, there are many neurons required for representing the best path way. [235] presents a coupled neural network, called output-threshold coupled neural network (OTCNN). However, this algorithm was very hard to be implemented in the real cases and it was improved in [153] by proposing a new modified model of Pulse-coupled neural networks (M-PCNNs). This model was improved by Liu et al by solving the K Shortest Paths (KSPs) problem [119]. In addition, the neural network based path planner can also be applied in non-holonomic mobile robot [227]. Furthermore, shunting based neural model first proposed by Hodgkin and Huxley [78] and refined by Grossberg in the neural mechanism [66] was also used to solve path planning model [226, 225]. However, the current neural based models [238, 226, 225, 119, 227, 111] have not considered rough terrain with undefined travel cost or weighting cost.

In this proposed research, we create a new model of dynamic path planning based on neuronal activity. A 4-legged robot is applied to prove the effectiveness of the proposed path planning model. The proposed model is expected to be online generated with obstacle avoidance and applied with undefined travel cost. These advantages do not exist in the current

state-of-the-art research. Nevertheless, an integrated system is required for supporting the proposed model.

The main contribution of the proposed research is to use the natural mechanism of the human brain for generating the online path planning in 3-D rough terrain with undefined travel cost. Our proposed reserach not only emphasizes the inner state process of the neuron, but also the development process of the neurons in the brain. Based on [29], the development of the brain involves the synaptic pruning process. The weak synapses will be deleted based on their synaptic efficacy and replaced by efficient synapses [30]. These processes are applied in this proposed path planning optimization process. Other contribution is to generate dynamic 3-D path planning with undefined synaptic weight. This proposed model let the neuron generate the signal based on the terrain condition in order to create the synaptic weight connection.

This chapter is organized as follows. In Section 7.2, we discuss the model of neuron mechanism that is used in this proposed path planning. Section 7.3 shows the design of the object application. In order to show the effectiveness of the proposed model, some experiments are presented in Section 7.4. Finally, Section 7.5 concludes the proposed research.

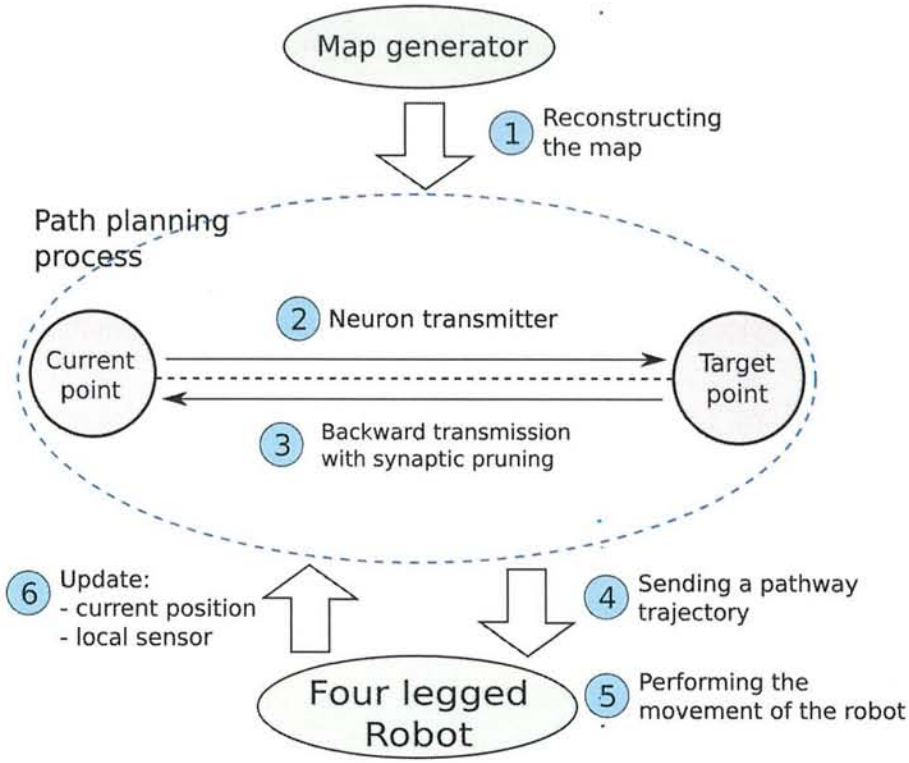



**Figure 7.1:** *Design of the proposed model*

## 7.2 The Proposed Model

The proposed model will be applied as the path planner in the robot movement on rough terrain. The full design of the system is depicted in Fig. 7.1. In the first step we generate the structure of the map by using Kinect sensor and send it to the proposed system by using radio communication. In the second and third steps, the proposed path planning are performed. After that the path planner sends the best pathway to the robot by using Bluetooth communication. In the fifth step, the robot moves and updates the current position and also updates some map information. The details will be explained in Section 7.3.

In this section we explain the novel neuro-activity based path planner. There are two algorithm processes in this proposed model, the forward transmission to construct the neuron connections for finding the possible way and the backward neuron transmission with synaptic pruning model for finding the best pathway from the current position to the target position and for reducing inefficient neurons. Dynamic path planning is also considered in this proposed model in order to respond to unpredictable obstacles. In the process mechanism, first the starting point neuron generates the signal to the neighbor neurons and calculates the synaptic weight value. After the signal triggers the target neuron, then the transmission process is stopped. After that, the backward signal process with optimizing the number of synapses is performed.

### 7.2.1 Neural Forward Transmission Based Model

The aim of forward transmission is to construct and build the neuron connections in order to find the possible way. In neural transmission studies, a signal is transmitted from one neuron to another neuron through electrical and chemical process (neurotransmission) occurring in the synapse. There are 2 types of synapse, electrical synapse (dendrite - dendrite) and chemical synapse (axon - dendrite). Chemical neurotransmitter requires releasing neurotransmitter into the synaptic cleft before a synaptic potential can be produced as input to the other cell. Membrane potentials are generated by chemical and electrical synapses function as a neuron's input.

The neuron typically has thousands input synapses and produces an action potential when the post-synaptic potential passes the threshold of membrane potential. Once there, it will trigger its own neurotransmitter release, which will cause a synaptic potential in the new post-synaptic neuron. The action potential will travel along the axon until it reaches the output (chemical) synapse at its axon terminal.

In the spiking model, there are timing models proposed for post-synaptic potential

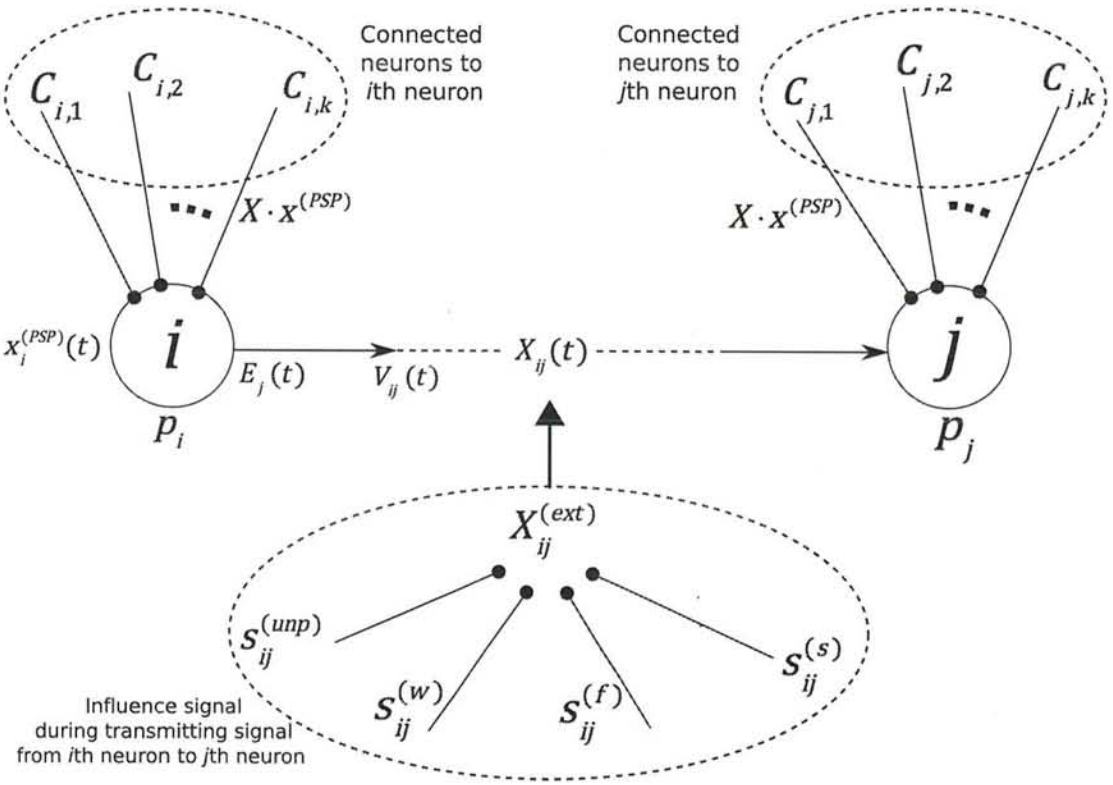



**Figure 7.2:** *Neuro-transmission model*

model [78, 122]. We use an exponential equation in order to model the decreasing energy. We modified and applied the neuron transmission activity in order to generate the best path planning in 3 dimensional rough terrain with unpredictable obstacle.

In this algorithm we use a forward transmission model depicted in Fig. 7.2. The strength of signal diversity from the $i$th neuron to the $j$th neuron is represented by $X_{ij}$. We assume, that the synaptic weights between each neuron are not defined. The strength of signal diversity is influenced by the potential energy in source neuron, the external influence between 2 neurons, and the internal disturbance. The strength of signal diversity decides whether the neuron to be connected or not as calculated in Eq. (7.1), where notation $A$ represents the inhibitory effect calculated in Eq. (7.2). The potential energy in the $i$th neuron is represented by $x_i^{(PSP)}(t)$ and the external influence between the $i$th and the $j$th neuron is represented by $X_{ij}^{(ext)}(t)$. $V_{ij}$ is the potential force from the $i$th neuron to the $j$th neuron.

In the transmission process, after certain neuron is activated (firing), the neuron is preparing the impulse signal, therefore the signal can reach the target neuron. If the neuron is already activated, then the preparation state to certain neuron is stopped. After the value of signal reaches the threshold of the target signal, then the firing process is performed. The neuron's capability for firing the signal is computed in Eq. (7.3), where $\mathbf{C}_i$ is a matrix representing the possible connection between the $i$th neuron and other neurons. In order to find the possible connection, we used a simple surrounding area algorithm which is computed in Eq. (7.10) and explained in Algorithm 7.2.

$$\dot{X}_{ij} = X_{ij}(t-1) + V_{ij}(t) - A \tag{7.1}$$

$$A = \sum_{j=1, j\neq i}^{N} X_{ij}(t-1) \cdot x_i^{(PSP)}(t-1) \tag{7.2}$$

In the initial condition, the strength of signal diversity ($\mathbf{X}$) and the potential energy ($\mathbf{x}^{(PSP)}$) are zero in each neuron and the time signal impulse ($\mathbf{t}^{(imp)}$) is 10000. When the path planning generation starts, the starting neuron is firing by the given parameter $t^{(imp)}$ representing the starting neuron as zero. It causes the parameter $\mathbf{x}^{(PSP)}(t)$ triggered, as calculated in Eq. (7.3), where $P_i^{(PSP)}$ is the maximum potential value in each neuron. After the neuron is fired, then the neuron generates the release energy $(E_i)$ depending on the derivative of potential energy degradation at time signal impulse $(dx_i^{(PSP)}/dt_i^{(imp)})$.

During the signal transferring process from the $i$th neuron to the $j$th neuron, signal strength $X_{ij}$ is supported by potential force $V_{ij}$ which is influenced by $h_{ij}^{(ext)}$, where $V_{ij}$ is calculated in Eq. (7.6). If the signal strength $X_{ij}$ is greater or equal than the distance between

2 neurons $\|\mathbf{r}_i - \mathbf{r}_j\|$ then the $j$th neuron is firing and $t_j^{(imp)} = 0$. If the signal strength is still lower than required, then $t_j^{(imp)}$ is increased and the potential energy is decreased. In order to acquire the travel time from the $i$th to the $j$th neuron, variable $T_{ij}$ is calculated in Eq. (7.4).

$$t_i^{(imp)} = \begin{cases} 0 & \text{if } X_{ij} \geq \|\mathbf{r}_i - \mathbf{r}_j\|, j \in \mathbf{C}_i \\ t_i^{(imp)} + 1 & \text{otherwise} \end{cases} \tag{7.3}$$

$$T_{ij} = \begin{cases} t_i^{(imp)} & \text{if } X_{ij} \leq \|\mathbf{r}_i - \mathbf{r}_j\|, j \in \mathbf{C}_i \end{cases} \tag{7.4}$$

$$x_i^{(PSP)}(t) = P_i^{(PSP)} \cdot e^{\left(\frac{-t_i^{(imp)}}{\tau_i}\right)} \tag{7.5}$$

$$\frac{dV_{ij}(t)}{dt} = V_{ij}(t-1) + \left(E_i - X_{ij}^{(ext)}(t)\right) \tag{7.6}$$

Figure 7.3 shows how the environmental condition influences the spike signal travel. Figure 7.3a illustrates the neuron connection from top view. Only the neuron inside the influenced area can influence the spike signal travel. In Fig. 7.3b, the height difference between 2 neurons causes the weight force ($s^{(w)}$) and the friction force ($s^{(f)}$) to also influence the travel of spike signal.

The external influence signal between the $i$th neuron and the $j$th neuron at certain time sampling $t$ is represented by $X_{ij}^{ext}(t)$ and computed in Eq. (7.7). There are 4 influence parameters calculated in Eq. (7.7), where $s_{ij}^{(unp)}$ represents the unpredictable obstacle, $s_{ij}^{(w)}$ represents the influence of weight force of the robot which is illustrated in Fig. 7.3, $s_{ij}^{(f)}$ is the constant value representing the predicted friction value, and $s_{ij}^{(s)}$ represents the influence of the other neurons against the signal between the $i$th neuron and the $j$th neuron. In Eq. (7.8), $s_{ij}^{(w)}$ is the external disturbance representing the self influence of the robot's weight between the height position of the $i$th neuron and the $j$th neuron depicted in Fig. 7.3b, where $m$ is the robot's mass and $g$ is the gravity acceleration. In Eq. (7.9), the effect of the height of the neuron around the signal line is calculated, where $d_{C_{j,k}} = (\mathbf{r}_j - \mathbf{x}_{i,j}^{(r)}) \cdot (\mathbf{r}_{C_{j,k}} - \mathbf{x}_{i,j}^{(r)})$, $h_{C_{j,k}} = (\mathbf{r}_{C_{j,k}} - \mathbf{x}_{i,j}^{(r)})[0\ 0\ 1]$, $\mathbf{x}_{i,j}^{(r)}$ is the location vector of the spike signal during neuron transmitting from the $i$th neuron to the $j$th neuron, and $d_{max}$ is a constant value representing the maximum influence of the neuron signal.

$$X_{ij}^{ext}(t) = s_{ij}^{(unp)} + s_{ij}^{(w)} + s_{ij}^{(f)} + s_{ij}^{(s)} \tag{7.7}$$

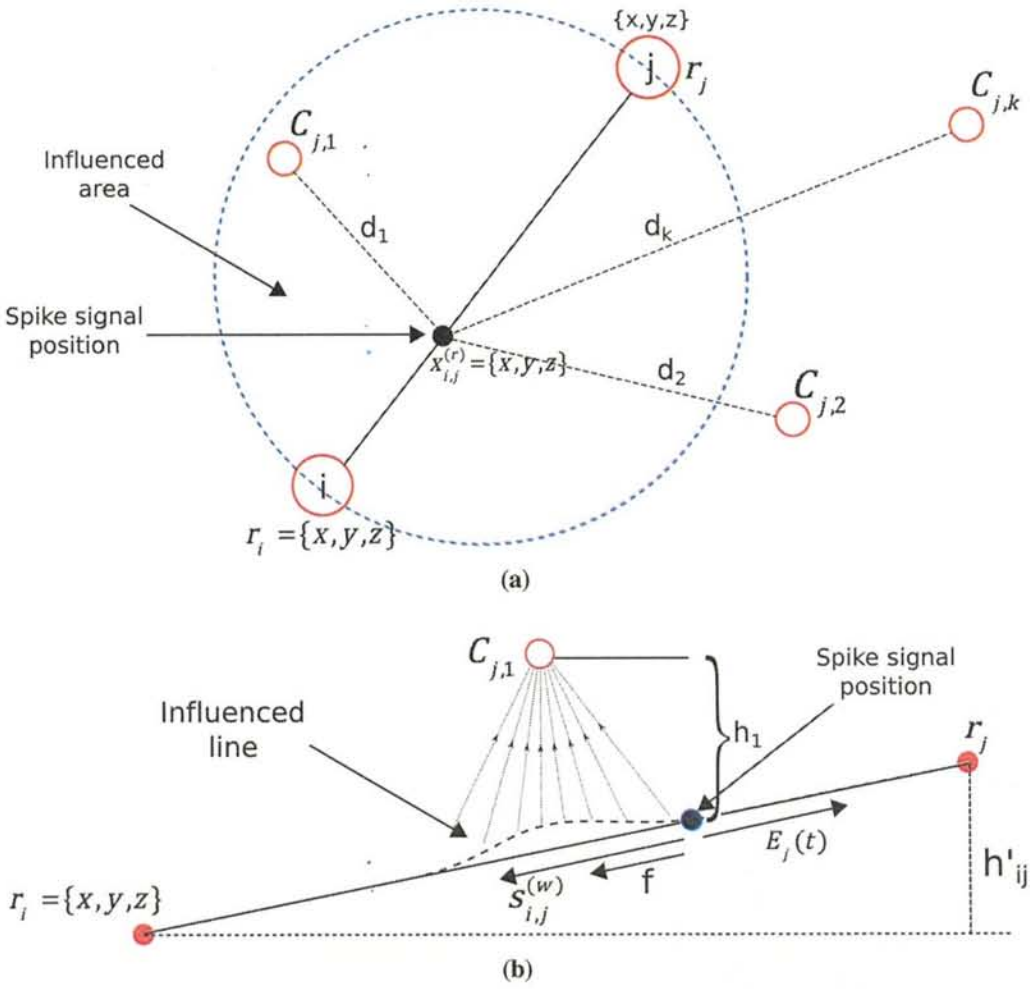



**Figure 7.3:** *Environmental influence represented by surrounding neurons, where each neuron has 3-D coordinate information (a) from top view (b) side view*

$$s_{ij}^{(w)} = m \cdot g \cdot [0\ 0\ 1] \frac{\mathbf{r}_i - \mathbf{r}_j}{\|\mathbf{r}_i - \mathbf{r}_j\|} \tag{7.8}$$

$$s_{ij}^{(s)} = \sum_{k=1}^{C_{j,0}} d_{C_{j,k}} \cdot h_{C_{j,k}} \cdot \exp\left( - \left( \frac{\|\mathbf{r}_{C_{j,k}} - \mathbf{x}_{i,j}^{(r)}\|}{d_{max}} \right) \right) \tag{7.9}$$

$h_{ij}^{ext}$ represents the external influence between the $i$th neuron and the $j$th neuron. $\mathbf{C}_i$ contains the neuron IDs which are possibly connected to the $i$th neuron. $\mathbf{C}_i$ is defined based on Eq. (7.10), where $\mathbf{C}_{i,0}$ is the number of possible connections to the $i$th neuron. The spreading of possible connections is influenced by $\gamma^{inf}$ representing the strength of scope.

$$\mathbf{C}_{i,k} = j, \quad \mathbf{C}_{i,0} = k \qquad \text{if } \|\mathbf{r}_i - \mathbf{r}_j\| \leq \gamma^{inf}, \quad j \in \mathbf{C}_i, \quad k++ \tag{7.10}$$

The main process of the proposed algorithm is explained in Algorithm 7.1, which starts with initializing all components and finding the possible interconnection in each neuron explained in Algorithm 7.2.

**Algorithm 7.1** Pathway generation represented by Fig. 7.2

```
Initialization;
for i ← 1 to N_max^(neuron) do
    Possible interconnection generation in ith neuron explained in Alg. 7.2
end for
for i ← 1 to N_max^(neuron) do
    for k ← 1 to C_{i,0} do
        j ← C_{i,k}
        if F_{i,j}(t) > 0 then
            calculate X_{i,j}^{ext}(t) by Eq. (7.7)
            calculate X_{i,j}(t) by Eq. (7.1)
            calculate T_{i,j}(t) by Eq. (7.4)
            if X_{i,j} ≥ ||r_i − r_j|| then
                ith neuron spikes
                t_j^(imp) = 0
            end if
        end if
    end for
end for
```

**Algorithm 7.2** Possible connection searching algorithm

```
input: i as neuron ID
C_{i,0} ← 0
k ← 0
for j ← 1 to N_max^(neuron) do
    if i ≠ j then
        d ← ||r_i − r_j||
        if d ≤ γ_inf then
            C_{i,(k+1)} ← j
            k++
        end if
    end if
end for
C_{i,0} ← k
```

### 7.2.2 Synaptic Pruning with Backward Transmission Model

In human brain development, during childhood, more than half of the synapses in the human brain are removed until puberty phase, which process is called synaptic pruning. In 1979, Huttenlocher showed the evidence for synaptic degradation in some areas of the human brain [80]. This process improves the performance of an associative memory network with limited synaptic resource. The strategy of the synaptic pruning is based on the weakness of the synapses [29].

In this grand system, we use communication system where the size of data causes the accuracy and the speed of transmission. In the proposed path planning, the number of synapses is based on the signal transmission process. If we have a big number of neurons representing the map, then the number of synapses will increase. Therefore, we will optimize the number of synapses representing the path planning trajectory by using the synaptic pruning mechanism.

In the mathematical model shown in Eqs. (7.11) and (7.12), we use neural regulation-driven synaptic modification (NRSM) first proposed by Chechik et al [30].

$$\mathbf{W}'(t+1) = \mathbf{W}(t) - \big(\mathbf{W}(t)\big)^{\alpha}\mu(t) \tag{7.11}$$

$$\mathbf{W}(t+1) = \mathbf{W}'(t+1)\frac{f_i^0}{f_i^t} \tag{7.12}$$

In this proposed model, notation $\mathbf{W}_i$ represents the age of the $i$th neuron which also represents the age of the weight connections to the $i$th neuron. In Eq. (7.11), $\mu(t)$ is the

sigmoid function $\text{sgn}(\eta - \epsilon)$ at time step $t$, where $\eta$ is the threshold value of the neuron, and synaptic reduction, $\epsilon$ is the degree of pruning calculated in Eq. (7.13), and power $\alpha$ defines the degradation dimension parameter chosen in the range $[0, 1]$. In Eq. (7.13), $\alpha^{(l)}$ and $\alpha^{(d)}$ are constant variables representing the degree of neuron distance effect and the deviation angle effect, respectively and $h_i^{PSP}(t)$ is the presynaptic spike output in the $i$th neuron. In this SP model, we set $f_i^0/f_i^t$ that represents the input field of the $i$th neuron at time step $t$ in Eq. (7.12) as one.

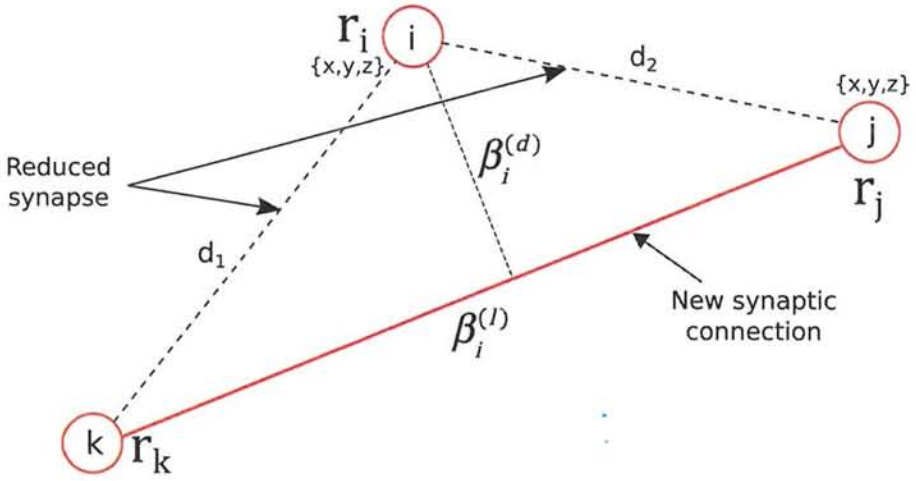



**Figure 7.4:** *Illustration of the components that influence the synaptic pruning*

$$\epsilon = h_i^{PSP}(t) \cdot \left(\alpha^{(l)}\beta_i^{(l)} + \alpha^{(d)}\beta_i^{(d)}\right) \tag{7.13}$$

$$\beta_i^{(l)} = \|\mathbf{r}_k - \mathbf{r}_j\| \tag{7.14}$$

$$\beta_i^{(d)} = \|(\mathbf{r}_k - \mathbf{r}_i) - ((\mathbf{r}_k - \mathbf{r}_i) \cdot \mathbf{n})\mathbf{n}\| \tag{7.15}$$

In this stage, we calculate the influence component in synaptic pruning process which is depicted in Fig. 7.4. We consider the length of the point $\beta_i^{(l)}$ calculated in Eq. (7.14) and the deviation angle $\beta_i^{(d)}$ calculated in Eq. (7.15) in the synaptic pruning model for the proposed path planning where its illustration is shown in Fig. 7.4. In the future we will consider the effect of the neighbor neuron in synaptic pruning. In Eq. (7.15), $\mathbf{n}$ is the unit vector of $(\mathbf{r}_k - \mathbf{r}_j)$.

The synaptic pruning model is performed during the backward signal processing. When the neuron is firing, then the SP decision is performed. When the age of the $i$th neuron $\mathbf{W}_i$ is getting zero then the connections to that neuron are reduced, and a new connection is created to the previous firing neuron that is connected to the $i$th neuron.

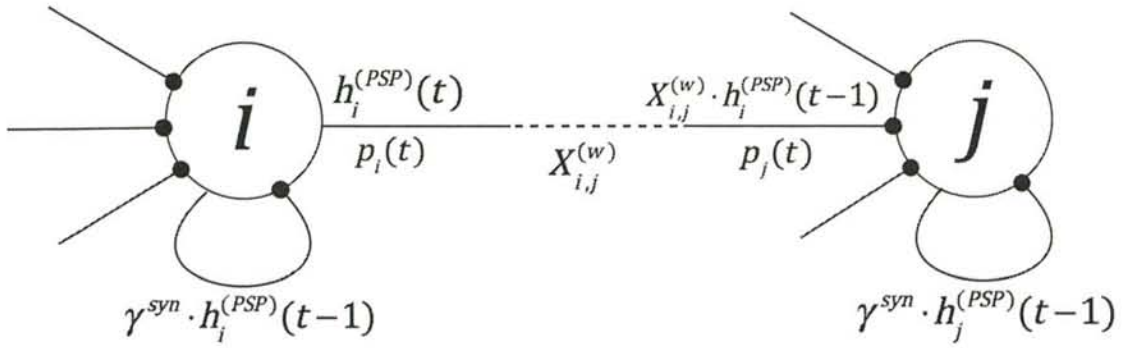



**Figure 7.5:** *Simple spiking neuron model for backward transmission*

In backward signal processing, we use a simple spiking neural network depicted in Fig. 7.5, whose mathematical model can be seen in Eqs. (7.17), (7.18), (7.19), (7.20). After the forward signal processing finishes, then the neuron at target point fires the impulse signal. The signal will be generated to the neuron at starting point. The backward signal will be calculated in Eqs. (7.17) and it is influenced by the synaptic weights calculated in the forward transmission. $h_i$ is calculated in Eq. (7.17), where $h_i^{syn}(t)$ includes the weighted pulse outputs from the other neurons which is calculated by Eq. (7.18) and the reduced value of the internal state in the previous time step which is calculated by Eq. (7.19).

In Eq. (7.18), $X_{ji}^{(w)}$ is a weight from the $j$th neuron to the $i$th neuron calculated from the travel time of neuron presented by variable $T_{ij}$ calculated in Eq. (7.4), $h_i^{PSP}(t)$ is the Post-Synaptic Potential (PSP) approximately transmitted from the $i$th neuron at the discrete time $t$, and $N$ is the number of neurons. The presynaptic spike output is transmitted to the connected neuron through the weight connection. $h_i^{PSP}(t)$ is calculated in Eq. (7.20), where $t_i^{(f)}$ is the last firing of the $i$th neuron, and $\tau_i$ is a parameter which influences how long the firing of the neuron has an effect to the connecting neurons. When the internal state (membrane potential) $h_i$ in the $i$th neuron is higher than the threshold $\Theta$, then the $i$th neuron fires the pulse signal calculated as follows:

$$p_i = \begin{cases} 1 & \text{if } h_i \geq \Theta \\ 0 & \text{0therwise} \end{cases} \tag{7.16}$$

$h_i^{ref}(t)$ is used for representing the refractoriness of the neuron. This means that after the neuron fires, its internal state value is decreased using the refractoriness component, in order to avoid the continuous firing of the neuron within a short time. In Eq. (7.19) $R > 0$, $\gamma^{ref}$ is a discount rate of $h_i^{ref}$, and $0 \leq \gamma^{ref} \leq 1$.

$$h_i(t) = \tanh(h_i^{syn}(t) + h_i^{ref}(t)) \tag{7.17}$$

$$h_i^{syn}(t) = \gamma^{syn} \cdot h_j^{(PSP)}(t) + \sum_{j=1, j\neq i}^{N} \mathbf{X}_{ij}^{(w)}(t+1) h_j^{(PSP)}(t) \tag{7.18}$$

$$h_i^{ref}(t) = \begin{cases} \gamma^{ref} \cdot h_i^{ref}(t-1) - R & \text{if } p_i(t-1) = 1 \\ \gamma^{ref} \cdot h_i^{ref}(t-1) & \text{otherwise} \end{cases} \tag{7.19}$$

$$\mathbf{h}_i^{(PSP)} = e^{\left(-\frac{t-t_i^{(f)}}{\tau_i}\right)}. \tag{7.20}$$

## 7.3 Application design

In this part we explain the integration system in order to support the proposed model. There are three supporting systems required for this proposed model, which are drone, server monitor, and robot. The drone is for generating the map and sending it to the server monitor via radio communication. It captures the area by using Kinect camera. The server monitor processes the proposed path planning based on the local condition that is sent from the robot and the primary map data sent from the drone. In order to find the unpredictable obstacle, the robot is equipped with ultra sonic (US) sensor and touch sensor, and periodically sent those sensor data to the server monitor via Bluetooth communication. The systems integration is depicted in Fig. 7.6.

### 7.3.1 Robot Design

In this research, we applied the proposed path planner model for walking trajectory generation of a 4-legged robot with 12 degrees of freedom depicted in Fig. 7.8 and its joint structure is depicted in Fig. 7.7, where there are 3 degrees of freedom (hip-x, hip-y, and knee joint) in each leg. The robot's size is approximately $30cm \times 18cm \times 20cm$, and its weight is 2 kg. Its mechanical frame is built using Bioloid Comprehensive Kit. The locomotion system and its stability have been developed in the previous research [178] combined with trajectory based locomotion [126, 174].

In order to support the proposed path planner and detect unpredictable collision, the robot is equipped with additional sensors, which are 4 touch sensors and 2 ultra sonic sensors.

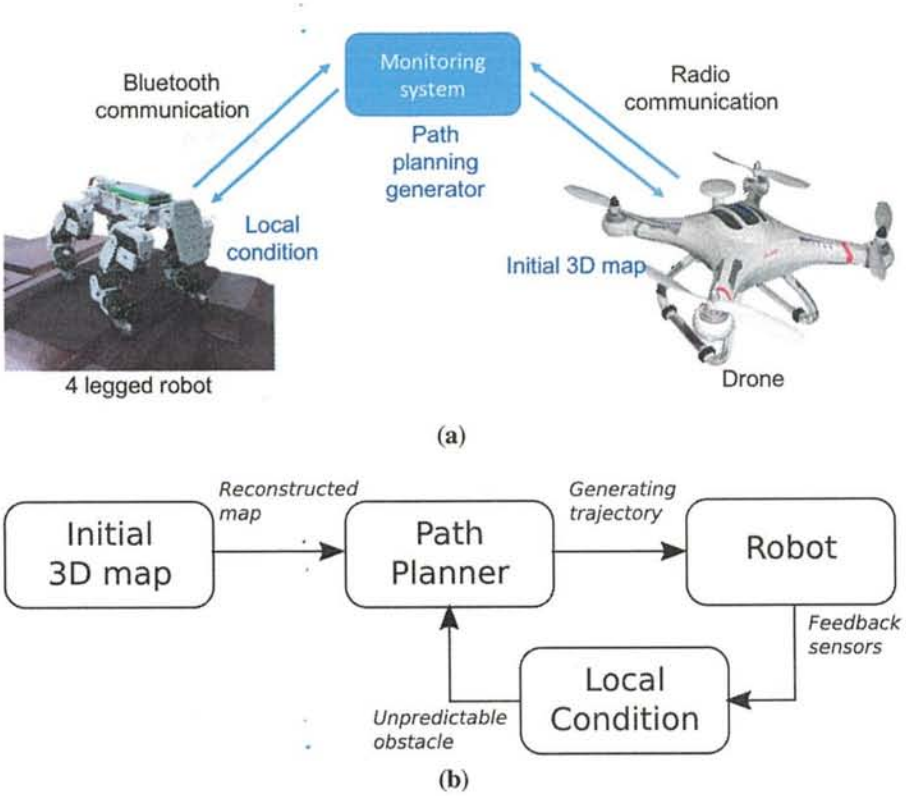



**Figure 7.6:** *Grand design of the integrated system (a) connection model (b) system model*

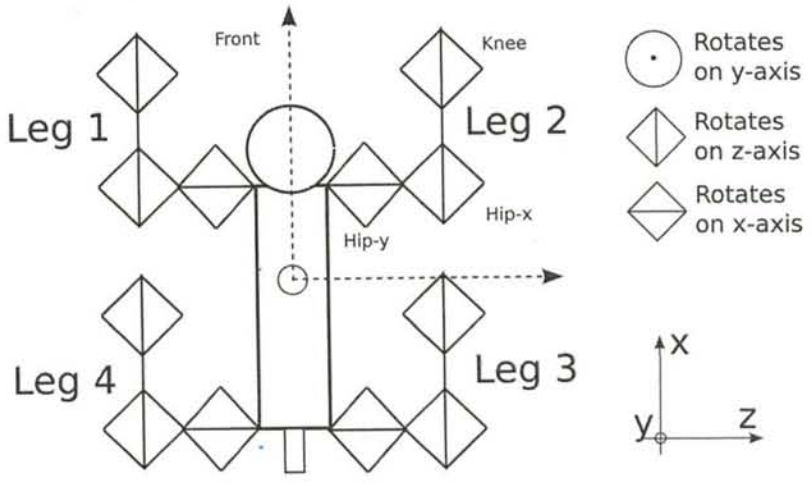
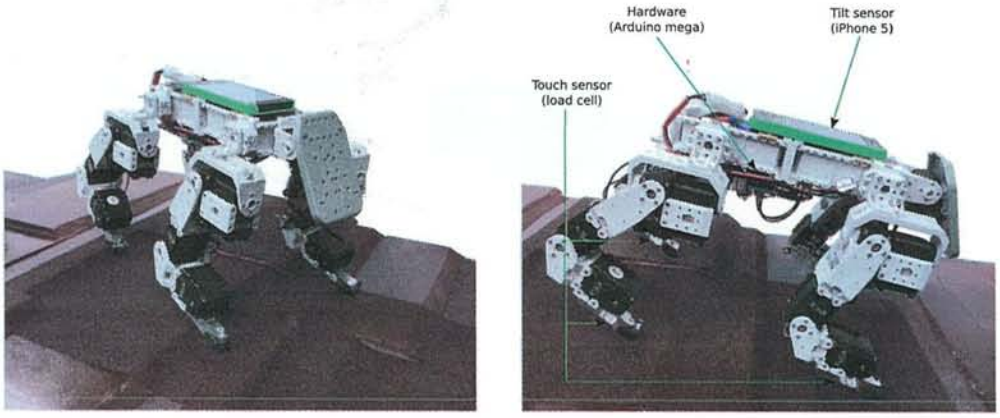



**Figure 7.7:** *Joint structure of the robot*




(a) (b)

**Figure 7.8:** *Design of the real robot a) from the front; b) from the left side*

### 7.3.2 Integrated System

In order to support the path planning algorithm in the 4-legged robot, a monitoring system and a drone are required in the grand design of this proposed system. Since the 4-legged robot supports the local condition of the robot, the drone supports the capturing of the initial data of map before the path planning is generated and the monitoring system processes the initial map, updates the local condition, and generates the path planning. Based on this integrated system, the possibility of the implementation of the dynamic path planning in disaster area is very high. The connection model of the supporting system is illustrated in Fig. 7.6a, where each system is integrated through wireless connection. In the grand application, we use Bluetooth connection for communicating between the robot and the monitoring system, and radio communication for communicating between the drone and the monitoring system.

## 7.4 Experimental Results

In the experimental result, we illustrate the effectiveness of the proposed neuro-activity based path planning model in both grid map model and topological map model. We show the signal transmission model and the synaptic pruning that is used in this proposed model. We create artificial terrain as the experimental object and show the effectiveness of the model responding to the environmental changing. In order to show the integration system, we applied the proposed model in a 4-legged robot movement trajectories in computer simulation ODE. Moreover, we implemented the proposed model in a real 4-legged robot, in order to show the feasibility in the real case.

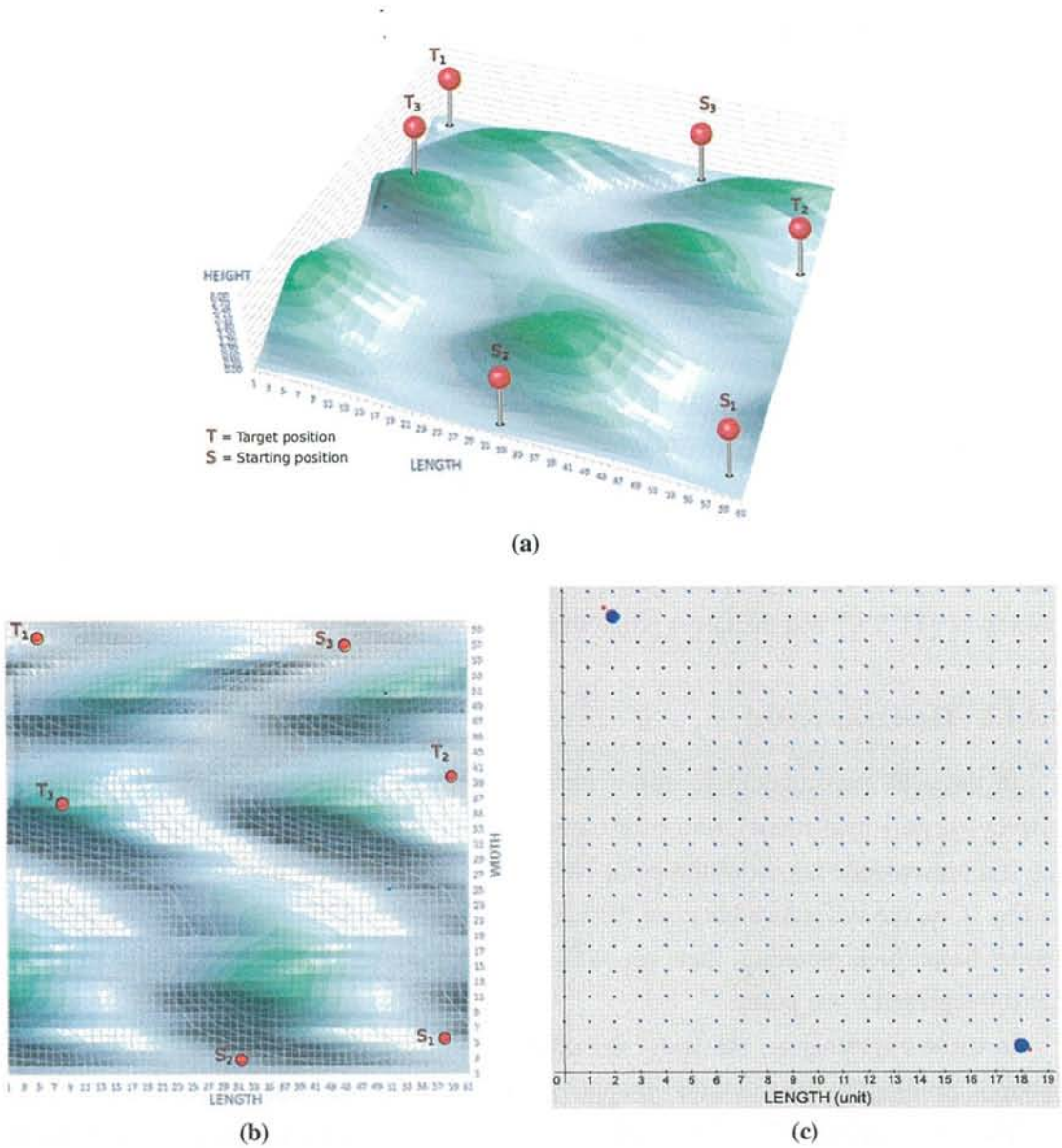



**Figure 7.9:** *Experimental terrain (a) perspective view (b) top view (c) Grid map model of experimental terrain*

**Table 7.1:** *Parameter Values*

| Parameter | $d_{max}$ | $P_i^{(PSP)}$ | $\gamma^{syn}$ | $\gamma^{ref}$ | $R$ | $\Theta$ | $\alpha^{(l)}$ | $\alpha^{(d)}$ |
|---|---|---|---|---|---|---|---|---|
| Value | 1.2 | 30 | 0.05 | 0.2 | 10 | 0.8 | 0.3 | 0.7 |

### 7.4.1 Neuron Transmission and Synaptic Pruning Experiment

In this experiment we create the illustration of the 3-dimensional uneven terrain. The artificial map can be seen in Fig. 7.9. In order to implement the proposed path planning, we converted the map data into grid map model and topological map. Furthermore, we compare the trajectory resulted by the proposed model. The parameters of FT and SP-BT model used in this experiment are shown in Table 7.1.

In this path planning we design the backward transmission with short term impulse firing signal. We put the same starting point and target point in grid map experiment and topological map experiment, both of them having 3 scenarios in this experiment with different starting point and target point which can be seen in Fig. 7.9. We choose the most difficult combinations of starting point and target point. In the first experiment, the starting point was $\{0.5, 11\}$ and the target point was $\{11.0, 1.0\}$, in the second experiment, the starting point was $\{0.2, 7\}$ and the target point was $\{6.5, 11.5\}$, and in the third experiment, the starting point was $\{11.2, 9\}$ and the target point was $\{7.0, 1.0\}$.

#### 7.4.1.1 Grid map model

In the grid map model, we adjust the resolution of the grid map model. Figure 7.9c displays a grid map model whose resolution is 20 x 20 unit. The different color in each point represents the height of the point. When the map is looked from the perspective view in Fig. 7.9a, the difference between the heights of each point can be seen. In this grid map model based path planning, there are 400 neurons representing the map model, since one neuron represents one point.

In these experiments, we show the difference between path planning model without synaptic pruning model and with synaptic pruning model as shown in Fig. 7.10.

The experiments in Fig. 7.10 confirm the effectiveness of the forward neuron transmission model in generating the possible pathway with undefined travel cost. The backward transmission also successfully formed the pathway based on the connection generated by forward neuron transmission.

From the result shown in Fig. 7.10, the path plannings using neuron transmission model with synaptic pruning model represent the path with smaller number of neurons. There are 25 neurons, 27 neurons, and 17 neurons representing the pathway without SP-BT model

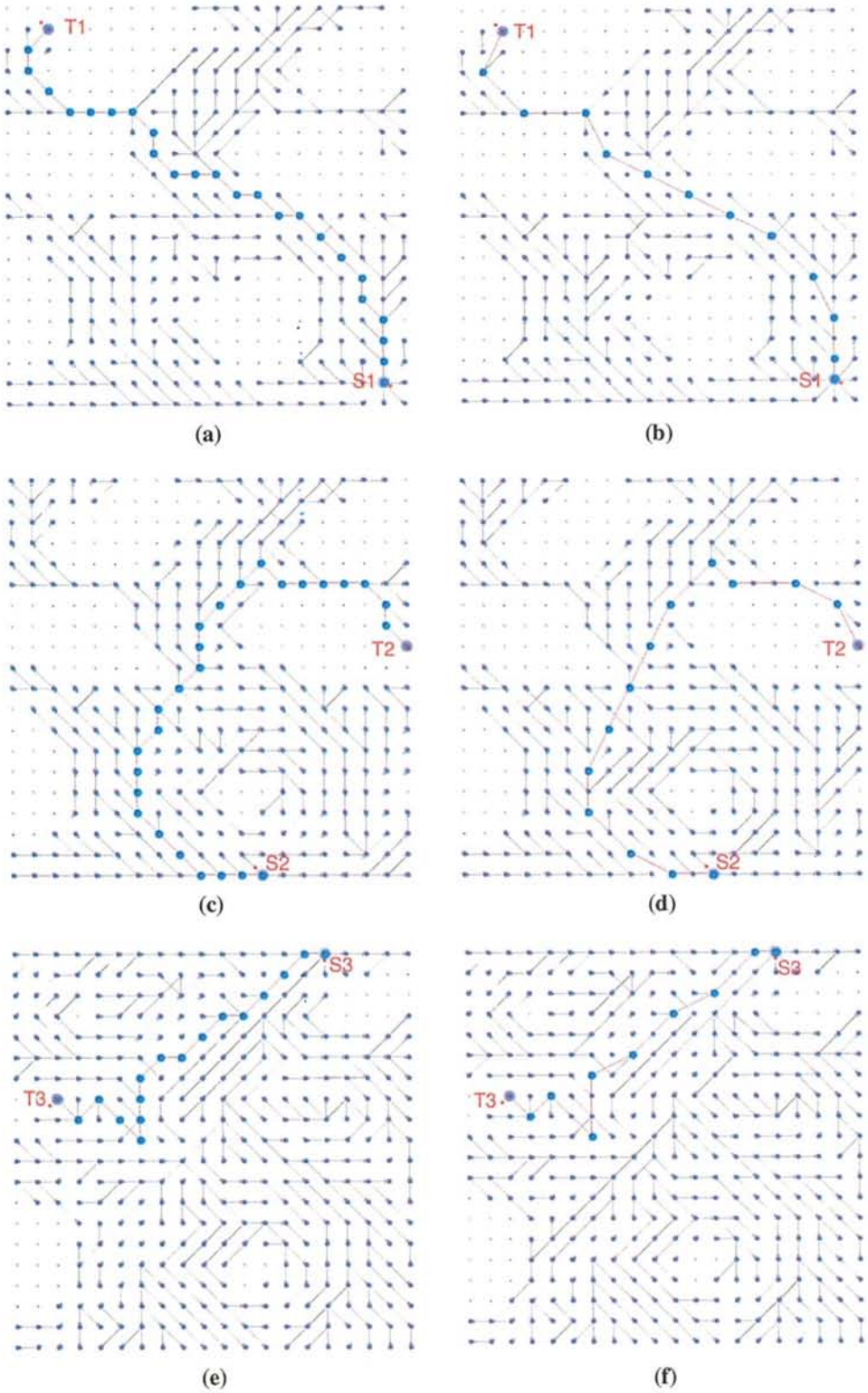



**Figure 7.10:** *Result of neuron transmission model in grid map model from top view (a) first experiment ($S1 = \{0.5, 11\}$ and $T1 = \{11.0, 1.0\}$) without SP-BT model (b) with SP model (c) second experiment ($S2 = \{0.2, 7\}$ and $T2 = \{6.5, 11.5\}$) without SP-BT model (d) with SP-BT model (e) third experiment ($S3 = \{11.2, 9\}$ and $T3 = \{7.0, 1.0\}$) without SP-BT model (f) with SP-BT model*

and 13 neurons, 14 neurons, and 10 neurons representing the pathway with SP-BT model in the first, second, and third experiment, respectively. SP-BT model proposed in this path planning model can effectively deactivate the weak neurons and also reduce the synapse which is connected to that neurons. In the case of huge map, this SP-BT model can reduce the size of the data representing the pathway therefore the communication weight can be reduced between the monitoring system and the robot.

#### 7.4.1.2 Topological map model

In the topological map based model, we reconstruct the map shown in Fig. 7.11 by using GNG proposed in [202]. By using this topological based map, the number of neurons required is smaller than in the grid map model. In the topological based map there are 165 neurons. In this experiment we set the maximum potential value $P^{(PSP)}$ as 45, 50, 60 in the first, second, and third experiment.

In these experiments shown in Fig. 7.12, we present the performance of the proposed path planning model. The SP-BT model was also applied in topological map model which is depicted in Fig. 7.12b. The SP-BT model reduced the number of evolved neurons in the pathway from 16 neurons to 8 neurons. The spike signal of neurons in this experiment can be seen in Fig. 7.13.

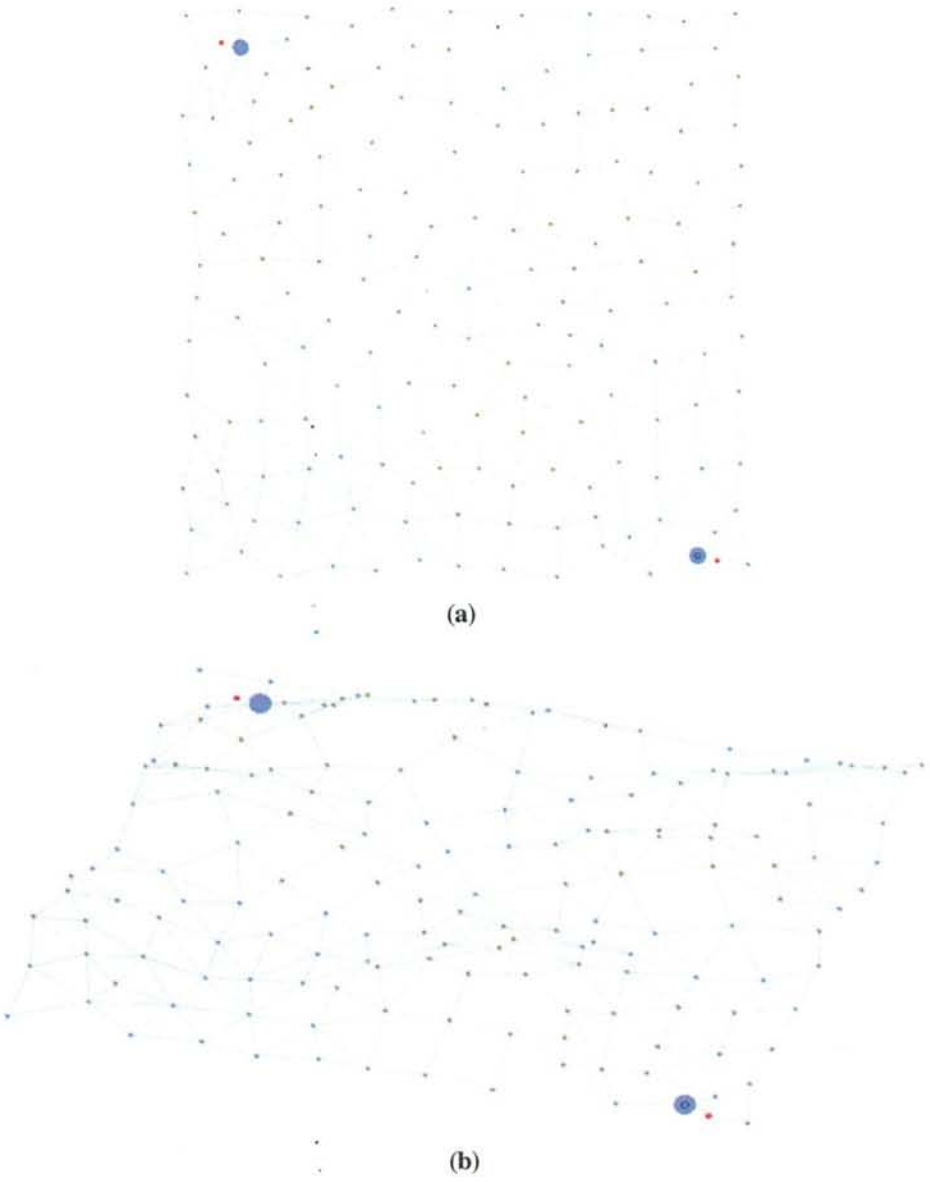
#### 7.4.1.3 Performed with unpredictable collisions

We put 3 unpredictable collisions in different positions, where the first, second, and third obstacle are located in $p_1 = (4.0; 9.0)$, $p_2 = (2.0; 10.0)$, and $p_3 = (7.0; 5.0)$, respectively. The result depicted in Fig. 7.10 shows, that the path planning model can dynamically change the pathway when it finds unpredictable collision. In Fig. 7.14a, the proposed model generates the initial pathway based on the initial map reconstruction. When the robot moves to a certain position, the robot finds the obstacle and the path planning model changes the pathway depicted in Figs. 7.14b, 7.14c, and 7.14d.

This experiment proves the flexibility of the proposed path planning model when finding unpredictable collisions.

## 7.5 System Integration

In this experiment we prove the effectiveness and the feasibility of the proposed model to be applied in the real cases. In order to be applied into the robot, it requires system integration for integrate all subsystem of motion capabilities. The system integration model can be seen



(a)

(b)

**Figure 7.11:** *Topological map model of experimental terrain (Fig. 7.9a) (a) top view (b) perspective view*

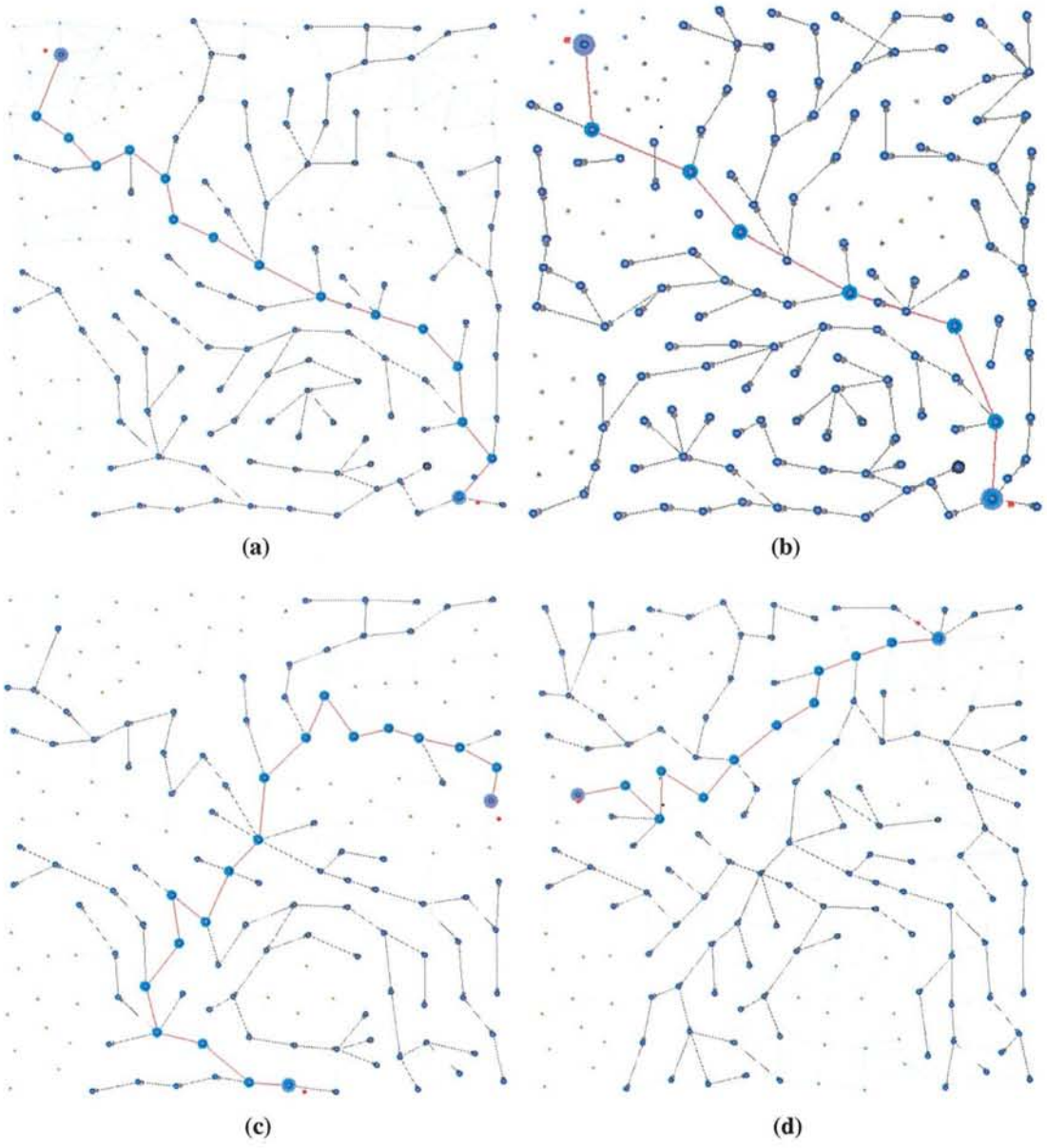



**Figure 7.12:** *Result of neuron transmission model in topological map model from top view (a) first experiment (b) first experiment with SP-BT model (c) second experiment (d) third experiment*

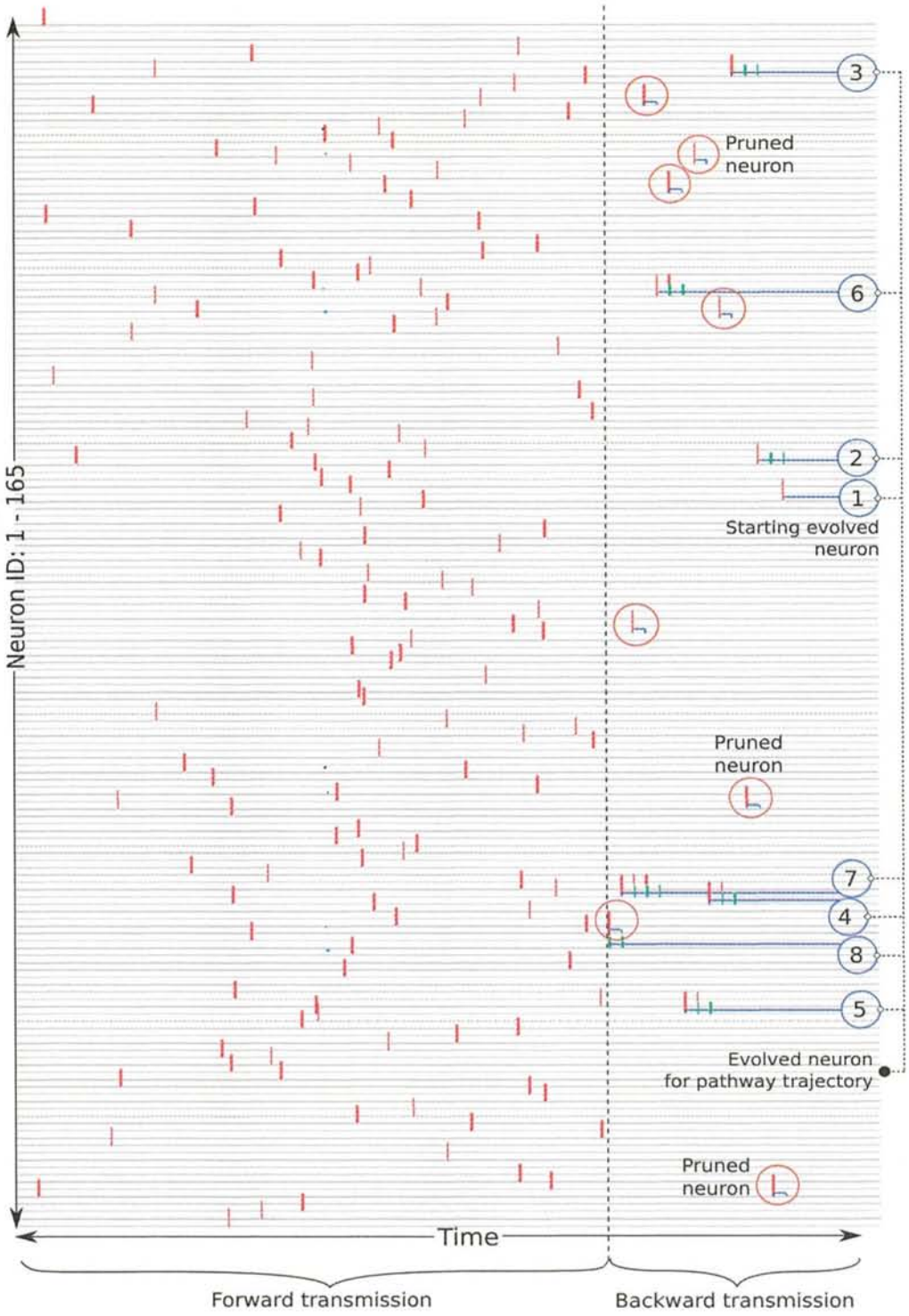



**Figure 7.13:** *Firing signal of neurons in FT and SP-BT processes, where the red circle represented pruned neuron and blue circle represented the neurons which constructed the pathway trajectory*

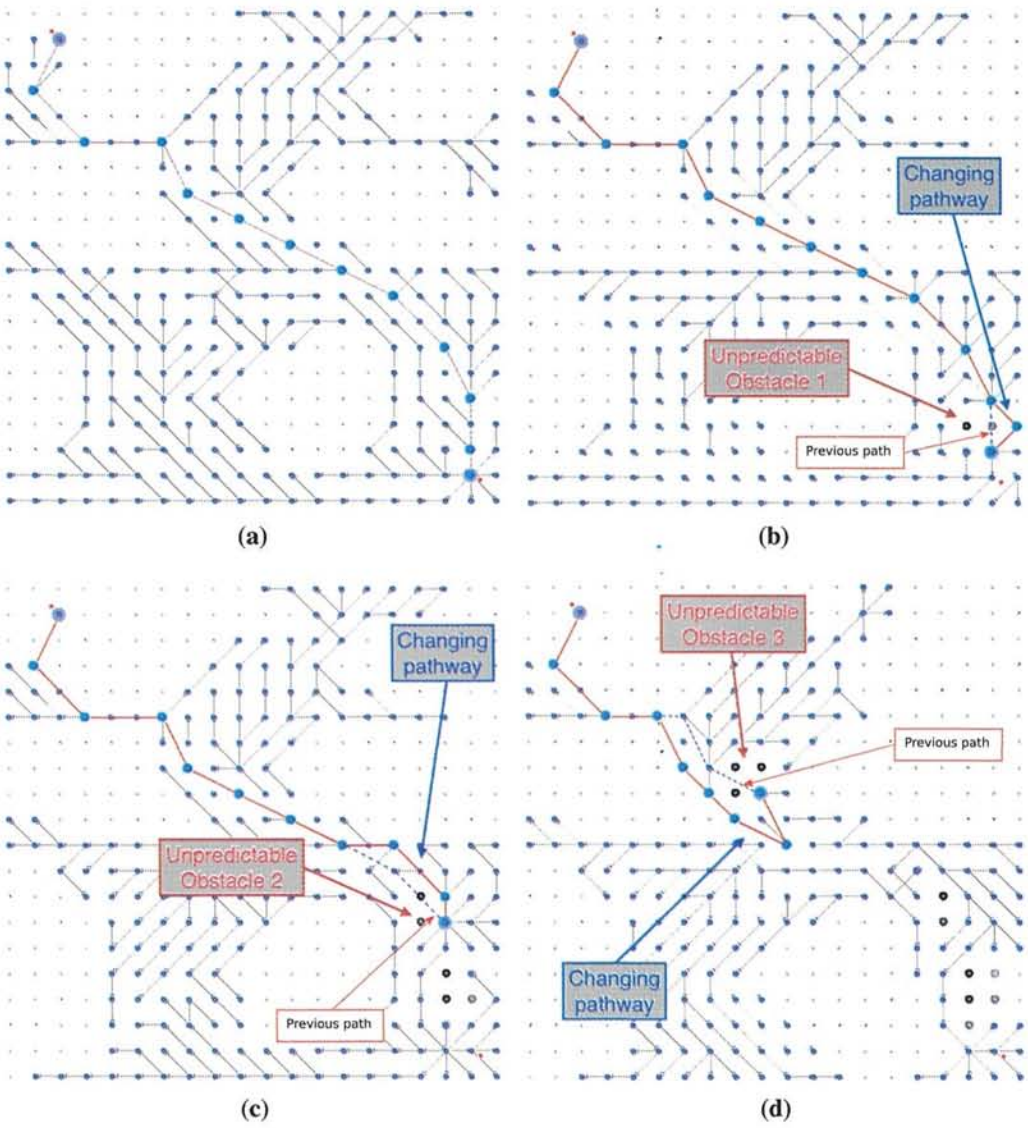



**Figure 7.14:** *Result of neuron transmission model in grid map model (a) initial pathway (b) changing pathway when the robot found the first obstacle (c) second obstacle (d) third obstacle. The black point represents the obstacle position and the gray point represents the influence of the existence of the obstacle*

in Fig. 7.15. There are 3 subsystem to be integrated, 1) motion planning model, 2) locomotion generator, and 3) stability model. From the topological map model, proposed neural based motion planning model generate path planning and send the desired walking provision to locomotion generator. After that locomotion generator generates signal in joint angle level. In order to recover and stabilize from disturabance, stability model sends feedback information to locomotion generator. These experiments are divided into 2 experiments, a simulation case and a real case.

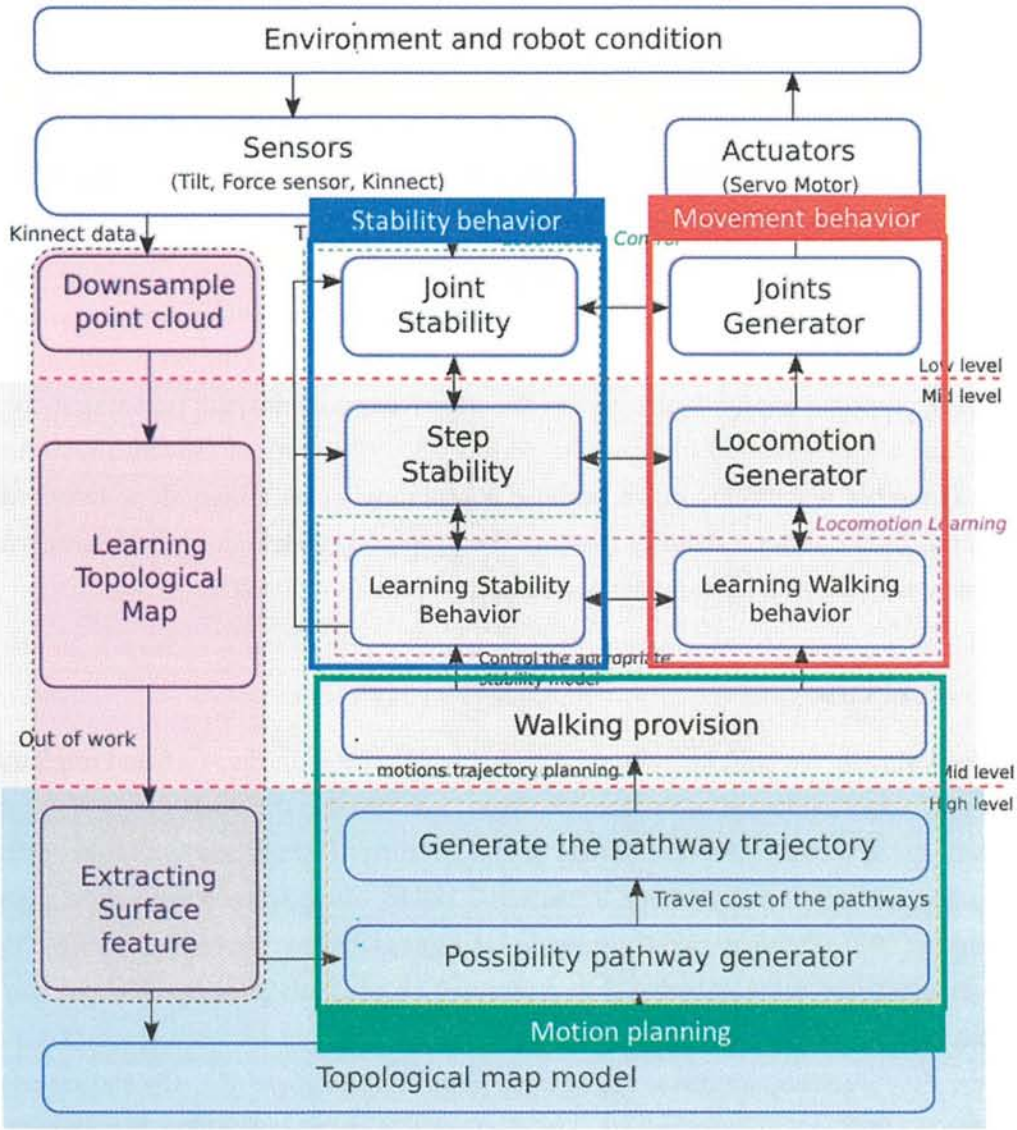



**Figure 7.15:** *Design of system integration*

### 7.5.1 Simulation case

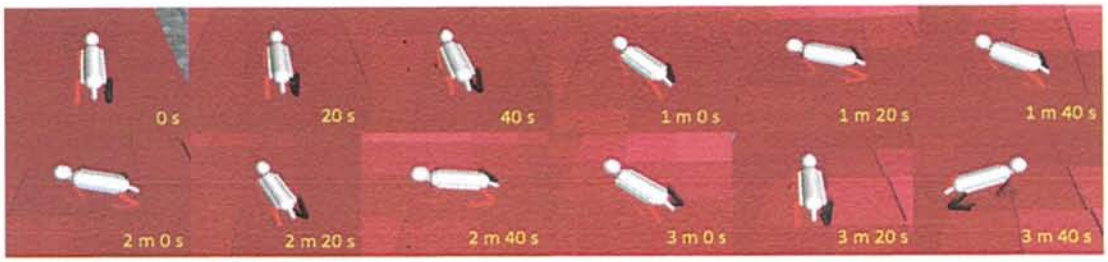
Open Dynamics Engine is used as the 3-D simulation software. We implemented the grid map model in order to adjust with simulation software ODE. The map model that is used for this experiment can be seen in Fig. 7.9c. In this simulation case, we perform the robot to move from starting point to target point based on the pathways generated in the Grid Map Model experiment. The result of the robot movement in each experiment can be seen in Fig. 7.17a and 7.17b. The result of the first experiment in the case without unpredictable obstacle shown from perspective view can be seen in Fig. 7.16. In the first experiment in the case with unpredictable obstacle, when the robot's sensor approaches and finds the obstacle, then the path planner dynamically changes the pathway shown in Fig. 7.17c. Figure 7.17c shows the effectiveness of the proposed model with unpredictable obstacles, where the pathway changes from the pathway shown in Fig. 7.17a.

When the robot moves directly from the starting point to the target point without following the pathway generated by the proposed path planning model, the robot will be falling down as shown in Fig. 7.17d. By following the pathway generated by the proposed path planning model, the robot successfully moved from certain starting point to target point. Figure 7.17 shows the comparison between robot movement with the proposed model and without the proposed model. It also proves the effectiveness of the path planning model that can generate a safe path and improve the performance of the robot movements on uneven terrain. The robot was falling down because it was moving on a high slope terrain that is not covered by the robot's stability system. The proposed model chose the fastest way but considering the safe pathway and the performance of the robot itself.

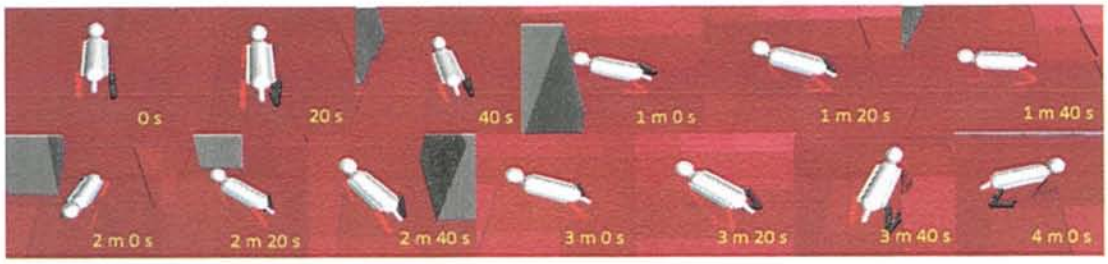
### 7.5.2 Real case

In the real case, we built the miniature of the map depicted in Fig. 7.18 and implemented the 4-legged robot explained in Section 7.3. We set low speed in the small 4-legged robot with maximum speed 5 cm/s. The size of map is 200 cm times 180 cm and it was reconstructed into topological map depicted in Fig. 7.18c and 7.18d by using 3-D reconstruction algorithm proposed in [202]. The dynamic environment is represented by the unpredictable obstacle. In this experiments, we conducted 2 scenarios which are path planning without and with unpredictable obstacle.

In the first scenario, we chose the starting point at coordinate (15, 15) and target point at coordinate (180, 15) in centimeter. The flexibility with any unpredictable obstacle was also performed in this experiment. We used the same starting and target point, and put the obstacle during performing the movement.




(a)




(b)

**Figure 7.16:** *Robot is performing the movement on rough terrain (first experiment) from perspective view (a) without unpredictable obstacle (b) with unpredictable obstacle*

The initial pathway (without obstacle) generated by the proposed model can be seen in Fig. 7.19a, when the robot detects the obstacle by using ultrasonic sensor, then the robot sends the information that there is an obstacle at certain position to the path planning system through Bluetooth communication. After that, the system changes to the other possible pathway. The changing pathway can be seen in Fig. 7.19b and 7.19c. By following the pathway generated by the proposed path planning model, the 4-legged robot successfully moved from certain starting point to target point and also successfully avoided the obstacle. This path planning model also improves the performance of the robot movement on uneven terrain as depicted in Figs. 7.20 and 7.21. Figures 7.20 and 7.21 show the robot movement trajectory on uneven terrain from perspective view and top view in both without unpredictable obstacle and with unpredictable obstacle.

In the parameter effect discussion, we analyze the adjusted parameter with the result of the pathway. Spreading parameter $(\gamma^{inf})$ representing the strength of scope in forward transmission model will affect the number of connections. If the parameter is too small, only few pathway possibilities can be generated by forward transmission. If the parameter is too big, inappropriate connection can also be generated. Therefore, this parameter should be adjusted according to the resolution of the topological map. Constant variables $(\alpha^l)$ and $(\alpha^d)$ representing the degree of neuron distance effect and the deviation angle effect in synaptic pruning model are the parameters that affect the reducing of the represented neuron in the

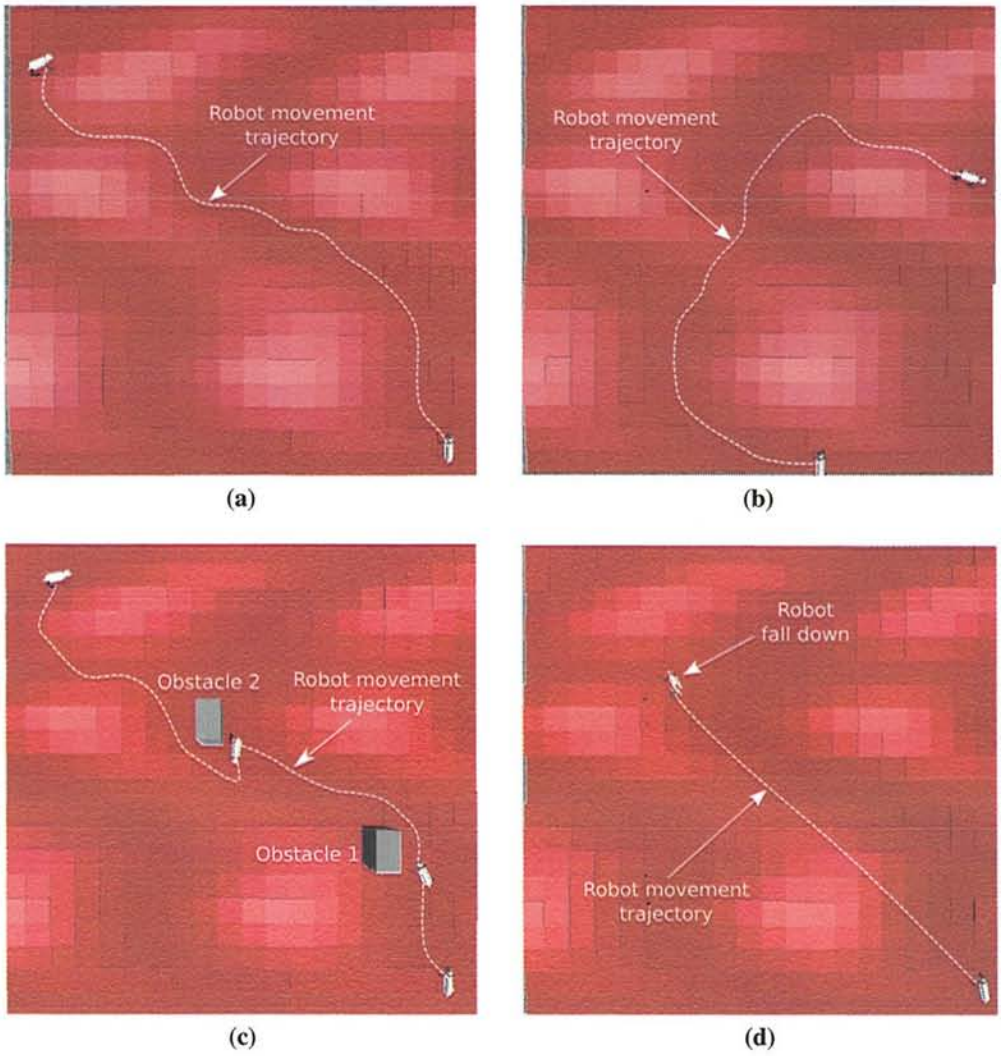



**Figure 7.17:** *Result of the robot movement based on the generated pathways (a) pathway generated in the first experiment (b) second experiment (c) first experiment with unpredictable obstacle (gray color as the obstacle and the black one is a shadow of the obstacle) (d) without performing the generated pathway (fallen down)*

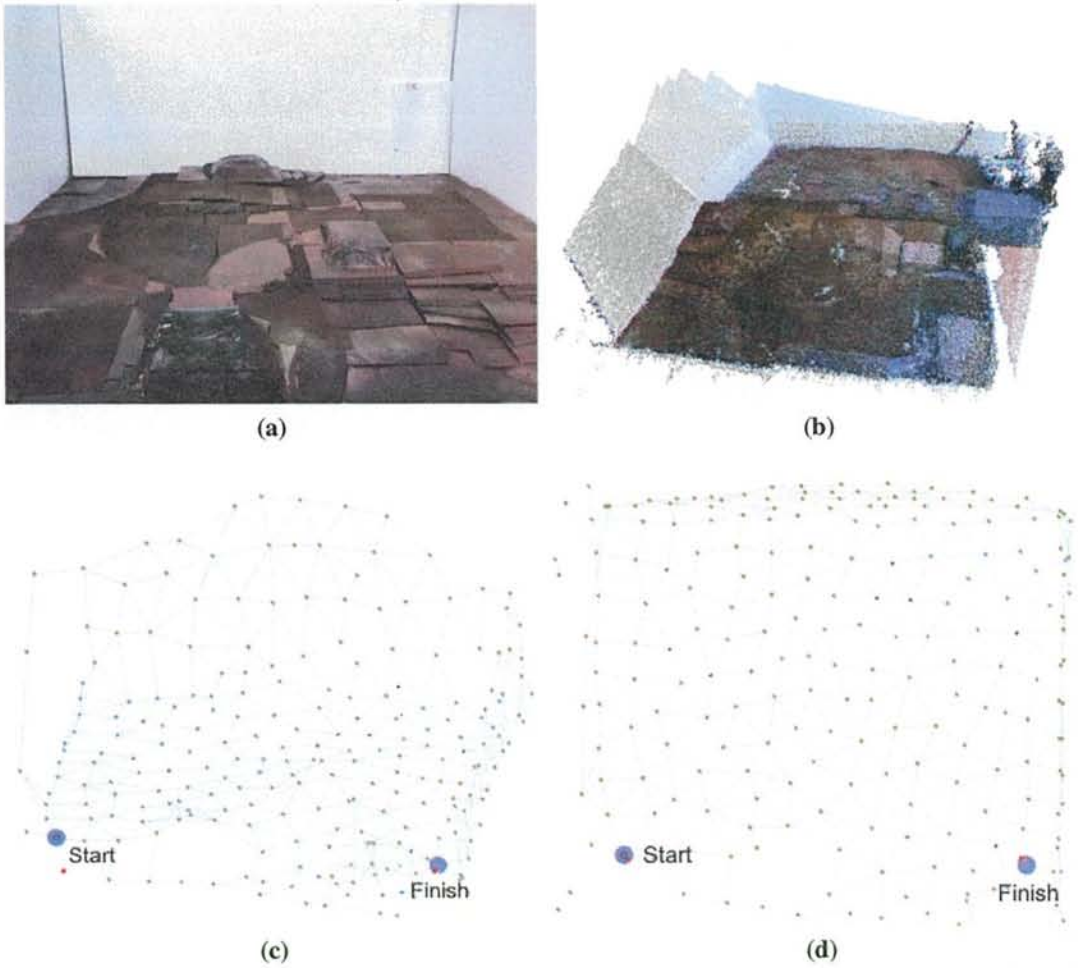



**Figure 7.18:** *Artificial map represents real cases (a) real terrain (b) feature extraction by using Kinect camera (c) reconstruction into topological map (d) topological map from top view*

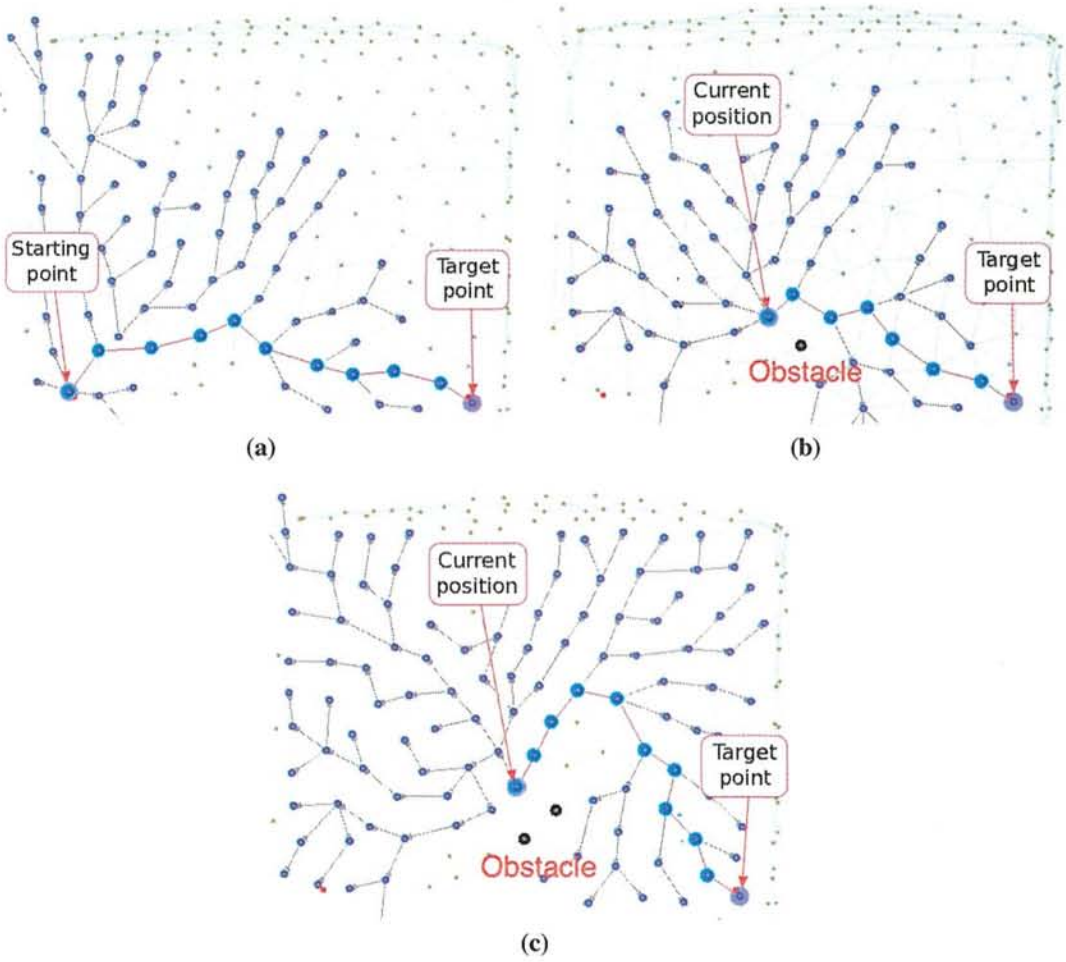



**Figure 7.19:** *Result of path planning model in topological map model (real terrain) (a) initial pathway (b) changing pathway when the robot found first obstacle (c) final pathway*

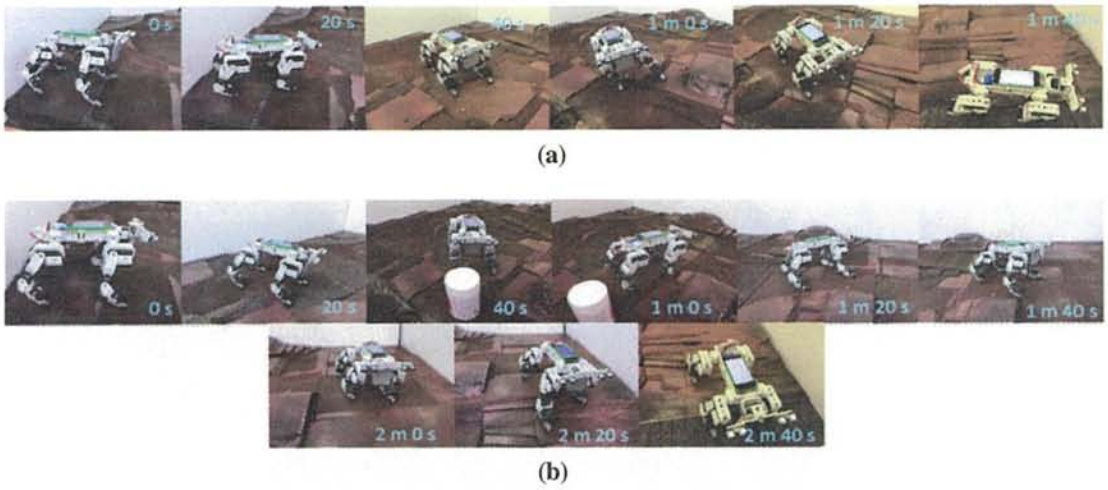



**Figure 7.20:** *4-legged robot is performing the movement on real rough terrain (a) without unpredictable obstacle (b) with unpredictable obstacle*

pathway. The value of parameters will influence the number of reduced neurons and the pathway changing. Therefore, the optimal value should be chosen based on the topological map resolution.

These implementations are still not enough for representing advance cases such as disaster area or mountains area. However, these implementations and the conducted experiments prove the capability of the proposed path planning model to be applied in further advance implementation.

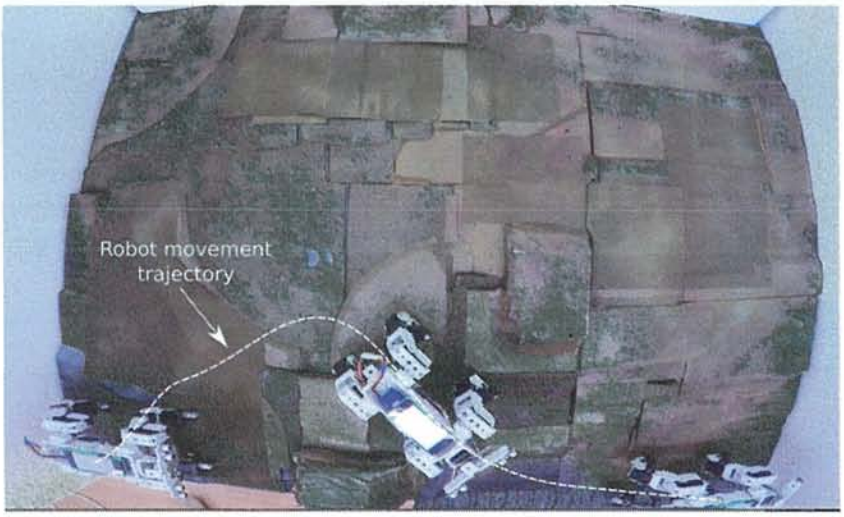
In the performance aspect analysis based on the experimental result, the proposed model has advantages which are listed as follows:

- Implementing online path planning with unpredictable obstacle.
- Considering 3-D path planning with undefined travel cost.
- Considering the neuron efficiency for reducing the memory usage in communication.

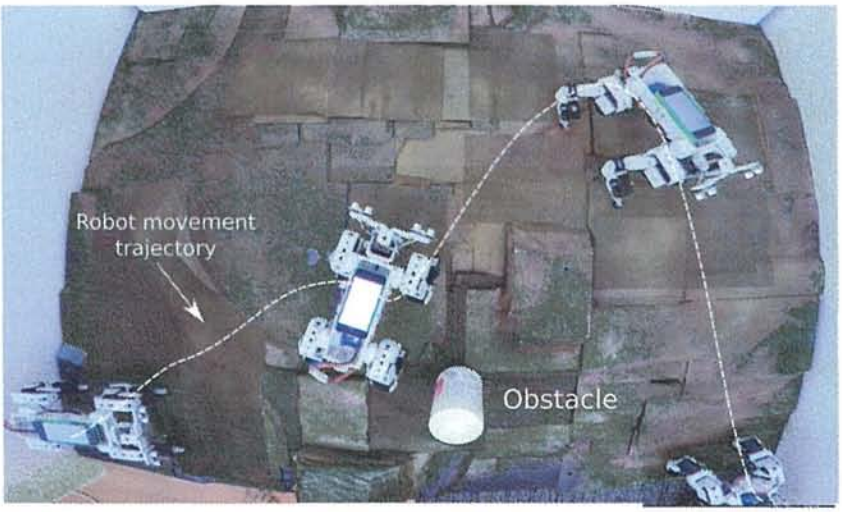
In this performance analysis, the computational cost is difficult to be compared, because the application and the constraint are different such as in 2-D or 3-D, the existence of travel cost, and its application area. Nevertheless, there are 6 aspects tabulated in Table 7.2 that shows the superiority of the proposed path planning model.

## 7.6 Discussion

This research proposed a natural mechanism of the human brain in order to generate a dynamic path planning in rough terrain with undefined travel cost. There are 2 algorithm




(a)




(b)

**Figure 7.21:** *Result of the 4-legged robot movement based on the generated pathways on real rough terrain (a) without unpredictable obstacle (b) with unpredictable obstacle*

**Table 7.2:** *Comparison of Path Planning Models and their Performances Aspects*

| Navigation Model | Dimension | Neuron Efficiency | Online/ Offline | Unpredict. obstacle | Travel cost | Application area |
|---|---|---|---|---|---|---|
| Modified D* [6] | 3D | - | Offline | - | Defined | Simulation |
| UAV 3D path planning [12] | 3D | - | Online | - | Undefined | Simulation |
| BMA based [13] | 2D | - | Offline | - | Undefined | Artificial environment |
| GNG map reconstruction [15] [16] | 3D | √ | Offline | - | - | Real environment |
| Modified D* [18] | 2D | √ | Offline | √ | Defined | Simulation |
| Neural based [26] | 2D | - | Online | √ | Defined | Simulation |
| PCNN: [27] [28] [29] [32] | 2D | - | Online | √ | Defined | Simulation |
| M-PCNN: [34] | 2D | - | Online | √ | Defined | Simulation |
| Proposed model | 3D | √ | Online | √ | Undefined | Simulation and Artificial environment |

processes in this proposed model, forward transmission and synaptic pruning with backward transmission. First, FT generates the impulse signal to neighbor neurons, defines the travel cost, and creates the neuron connection. Second, after target neuron is connected and gets the impulse, then the synaptic pruning and backward transmission are performed. These algorithms are implemented in grid map model and topological map model. One point is represented by a neuron in this proposed model.

Based on the experiments in both grid map model and topological map model, FT model successfully constructed the neuron connections for finding the possible way. SP with BT also successfully found the possible pathway from current position to target position and reduced inefficient neuron in both map models, however the considered condition in SP model is required to be increased. These possible pathways are also proved by performing the robot in both computer simulation and real robot. The robot successfully moved from starting position and target position when following the pathway generated by the proposed path planning. This proposed path planning model supports the performance of the robot movement. When the robot moved directly from the starting point to the target point without following the pathway generated by the proposed path planning model, the robot will fall

down.

The flexibility of this dynamic path planning was proved by the experimental result in computer simulation and real robot. When the robot follows the pathway and finds the obstacles, then the path planning model is changed to another possible pathway which is appropriate with robot ability.

In the future application, the proposed path planning will be applied for advance cases such as in disaster area, hills or mountains area. In order to improve the flexibility of the path planning, the supporting sensor for this proposed model will be improved in the robot application.

# Chapter 8

# Conclusion and Future Works

## 8.1 Conclusion

This proposed research presents the novelty and the development of motion capabilities of legged robot based on neural and natural process. There are three subsystems be developed and integrated, (1) locomotion behavior model, (2) stability behavior model (3) motion planning model.

In locomotion behavior model, there are several stage of developments which are dependent to each other. In the first development, we develop locomotion model which consists of four types of neurons, 1) motor neuron, 2) sensory neuron, 3) command neuron, and 4) gain neuron. Single objective and Multiobjective evolutionary algorithms are used effectively to construct the locomotion pattern by manipulating the weight of synapse between the motor neurons. However, multi-objective may generate more than one solution. Based on the result from the evolutionary algorithm, the model acquires the optimum solution as the fastest walking speed of the robot, according to its capability, with low inertial oscillation. In the proposed locomotion model, the control system for walking speed and walking direction is solved by determining the parameter of the command neuron signal transmitted, as in our brain, to the gain neuron. The output from the coupled neuron oscillator is controlled by the gain neuron, depending on the signal from the command neuron. In the real robot experiment, various walking directions and walking speeds were implemented as the result of the proposed system. This research also presented a new stability model that supports the locomotion system. The inertial sensor and ground touch sensor are installed to acquire the value of the sensory neuron. We determine the synaptic weight between the sensory neuron and the motor neuron dynamically by using RNN. In the experiment, the synapse weight between the sensory neuron and the motor neuron can be dynamically changed, depending

on the environmental condition. However, the neural structure is still static which causes the locomotion not perfectly dynamic.

In second development, we develop learning model of dynamic neural structure for controlling walking speed. We use evolutionary algorithm for forming several walking patterns that implies several interconnection structures with different walking speeds. The evolutionary algorithm with the proposed strategy effectively forms the walking pattern based on biological approach. In order to acquire the dynamical relationship between walking speed and its pattern, MLP is effectively implemented. The trained weight parameters in MLP are used for generating dynamical walking pattern by defining the desired walking speed in the gait generator system. By using the proposed locomotion model, the walking speed can be easily controlled. The locomotion generator can generate signal neuron transition smoothly with the change in walking speed.

In order to improve the performance and the functionalities, in third development, we develop the learning strategy for solving the complexity of neural structure in neural oscillator based locomotion for generating omni-directional movement behavior. Since multi-objective evolutionary algorithm is used as the behavior optimization, movement provision and resulted energy are calculated as the fitness calculation, desired movement behavior can be acquired with minimum energy required. In this optimization, the behavior solutions are not deserved well, however, by increasing the number of generation, good diversity can be acquired. In this proposed method, walking behavior references generation took a long time, one reference behavior requires one optimization process. This is one of the problems which should be solved in next development. Walking behavior references is used as the training data of the learning model. Separated MLP strategy is successfully approaching the relationship between input and output of walking behavior. Training iteration and learning process take a long time, and also require higher performance of learning model, such deep learning model. Input behavior references are also required to be increased in order to specify variant behavior in one movement provision, since in this current state, redundant behavior may be generated in the walking reference optimization. However, by using this proposed learning model with cross validation, omni-directional movement behavior can be generated. The model can generate sagittal movement from 0 m/s - 25 m/s, coronal movement from -5 m/s - 5 m/s, and turning speed $(-\pi/2)^0/s$ - $(\pi/2)^0/s$.

Since there is still ineffective neural model which cause the computational cost, in fourth development, we propose a new model of neuro-locomotion by implementing passive neural control represented by Bezier curve model. This model is implemented for quadruped robot. There are 4 neurons in the neural structure where one neuron represents 3 joints in one leg. Recorded joint trajectory of cat-like robot is required as reference of joints. In order

to optimize the relationship between signal generated by neuron and each joint, Bezier based optimization is implemented which has successfully established the relationship equation to generate the angle joint values with neural signal as input. Neural oscillator is optimized by evolutionary algorithm and is able to generate signal with $\pi/2$ phase difference that is suitable for walking pattern of cat. Phase difference can be different depending on the feedback signal from sensory neuron. This proposed system can reduce the neural complexity and also significantly reduce the computational cost by 63% as weIl. Therefore, this proposed neuro-Iocomotion system has successfully been applied to the small 4 legged robot with low frequency microcontroller and considers the influence of sensory feedback.

In order to improve the stability level, independent stability model is required to be developed. We develop the new bio-inspired stability system model. We develop MRNN as the online learning model. MRNN consist several RNNs where the number of RNNs is dependent on the complexity cases. There is selector system that decides certain RNN system depending the current condition of the robot. If the robot often gets the certain condition, then the RNN representing that certain condition has good response for the robot. There are 2 methods for placing the range of each RNN, first, placing the desire angle point inside the RNN range and second, placing the desire angle point in the RNN barrier. There are influences of number of RNN in the first placing method. The large number of RNN indicates better stability level which has small oscillation amplitude. Single RNN is not enough to stabilize the proposed case. MRNN gives better result than single RNN. In the second placing method there is no big different between small and big number of RNNs. However, big number of RNNs is required in order to learn multi-desire angle. The proposed model also successfully stabilized the robot from the disturbance in both placing methods. If the robot frequently experiences a certain condition, the stabilization response is getting better. Since the proposed stability model successfully applied in 2-wheeled robot.

In the next development, we evolve the proposed stability model as push recovery model in humanoid robot. MRNN that contains several RNNs are proposed for stability learning systems (SLSs) in three motion response behaviors, which are ankle, hip, and step behavior. One behavior is represented by one MRNN. MRNN successfully creates stabilized behavior under external disturbance. However further research is required to increase the number of considered output value. Push recovery controller is proposed for managing an appropriate motion behavior in order to respond to certain push disturbance. Online learning algorithm applied in push recovery controller based on MARS model successfully forms and decides appropriate motion behavior after a hundred given experiences perturbation with random force value. Experimental result also shows the effectiveness of the model stabilizing the robot walking under external perturbation. Based on that, push recovery controller is effec-

tive to be applied in humanoid robot as the online stability learning system.

When the inner motion capabilities have been well developed, the robot is required the guidance for moving to one place to another place. Therefore, in this stage of research, we develop path planning model with undefined travel cost constraint which is dynamic with unpredictable obstacle. This research proposes a natural mechanism of the human brain in order to generate a dynamic path planning in rough terrain with undefined travel cost. There are 2 algorithm processes in this proposed model, forward transmission and synaptic pruning with backward transmission. First, FT generates the impulse signal to neighbor neurons, defines the travel cost, and creates the neuron connection. Second, after target neuron is connected and gets the impulse, then the synaptic pruning and backward transmission are performed. These algorithms are implemented in grid map model and topological map model. One point is represented by a neuron in this proposed model.

Based on the experiments in both grid map model and topological map model, FT model successfully constructs the neuron connections for finding the possible way. SP with BT also successfully finds the possible pathway from current position to target position and reduce inefficient neuron in both map models, however the considered condition in SP model is required to be increased. These possible pathways are also proved by performing the robot in both computer simulation and real robot. The robot successfully moves from starting position and target position when following the pathway generated by the proposed path planning. This proposed path planning model supports the performance of the robot movement. When the robot moves directly from the starting point to the target point without following the pathway generated by the proposed path planning model, the robot will fall down.

The flexibility of this dynamic path planning is proved by the experimental result in computer simulation and real robot. When the robot follows the pathway and finds the obstacles, then the path planning model is changed to another possible pathway which is appropriate with robot ability. In this experiment, all subsystem is integrated into one interconnected motion capabilities model. In this implementation, all subsystem were analyzed and performed well and the robot able to stop in the goal point. These implementation proved the effectiveness of the system integration, the motion planning model is able to generate safe path planning, the locomotion model is able to generate flexible movement depending on the walking provision from motion planing model, and the stability model can stabilize the robot on rough terrain,

For summarizing the conclusion every development, we list several achieved contribution proved by proposed experiments.

- Musculoskeletal model represented by flexor and extensor muscle as the locomotion model is able to improve the dynamic factor in locomotion, therefore, it can generate

locomotion behavior

- Proposed learning strategy for neural oscillator interconnection model successfully generate omni-directional motion behavior.
- Multi-objective evolutionary algorithm can generate multiple solution of interconnection structure of motor neurons and sensory-motor neuron for generating locomotion behavior in uneven terrain.
- Proposed multimodal learning system for stability which implement combined Recurrent NNs (MRNN) successfully stabilize the given problems.
- Combined with MARS, the stability controller model is able to efficiently respond the external perturbation in biped and quadruped robot.
- The proposed natural mechanism of the human brain can generate the online path planning in 3-D rough terrain with unpredictable travel cost proved by proposed experiments.
- The forward transmission for constructing the neuron connections can find the possible way.
- The backward neuron transmission with synaptic pruning model can find the best pathway from the current position to the target position and reduce inefficient neurons.
- System integration succcessfully integrate three systems of motion capabilities that able to share the information in order to perform autonomous movement.

Generally, the proposed model can be expected to bring a great contribution to the motion capabilities development and can be used as alternative model for acquiring the dynamism and efficient model in the future instead of conventional model usage. This propose model has superiority in high number of behavior than conventional model. Since the conventional model should plan and build every behavior, the proposed model can be automatically generate the appropriate behavior.

## 8.2 Future Works

Specifically, in locomotion behavior model, variant of walking input references should be increased by considering internal and external sensory information in order to classify redundant behavior. In order to improve the performance of learning model, high performance deep learning with online application may be implemented.

In stability model, since the recovery behavior is still defined under preliminary test, adaptive recovery behavior should be developed so that the system can be easily applied into variant robots.

In neural based path planner, the proposed path planning is required be applied for advance cases such as in disaster area, hills or mountains area. In order to improve the flexibility of the path planning, the supporting sensor for this proposed model will be improved in the robot application.

In the previous research, generally, dynamic neuro-locomotion in humanoid robot and its stability have been well developed, in biped or quadruped mode that reduced the computational cost. Stability and omni-directional dynamic walking controller were considered in previous model by taking internal sensor information. However, the previous model is limited to internal dynamic. Therefore, it has some limitations for achieving environmental perception, movement prediction, planning, and energy efficiency. Senso-ry-motor coordination is one of the most important research topics to deal with such kind of adaptive locomotion. Further-more, the locomotion itself also has complex dynamics. To summarize, we have to deal with (1) external dynamics in en-vironmental changes (2) external dynamics in human behaviors, and (3) internal dynamics in locomotion, simultaneously.

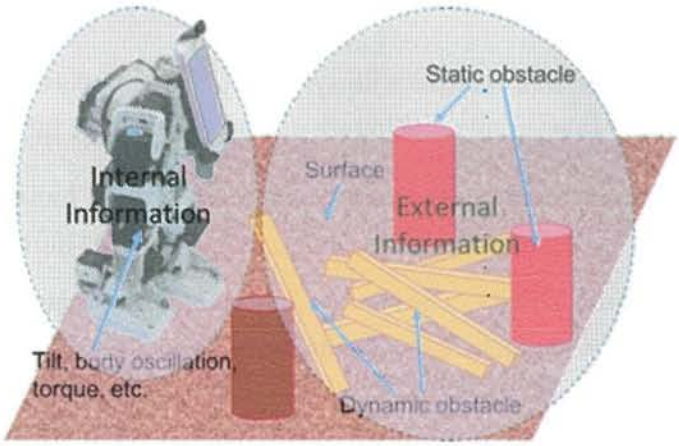



**Figure 8.1:** *Diagram of Internal and external sensory information*

According to R. Preifer, cognition, embodiment structure, and locomotion generator correlation should be integrated for developing solid methodology in neurobiological locomotion model. [151]. Human and animal's gaits correspond to some particular physical system, which the movement is natural with respect to their body and it requires minimal energy and little control. In this current issue, most of researchers conducting locomotion study only process on either the internal or external sensory information separately. They did not consider the integration of cognition, locomotion, and embodiment (3 Aspects). However, bridging

connection between internal and external information will be affect-ing the computational cost. In multiple gait generators, researchers separate multiple gaits into few locomotion models. By separating multiple gaits, there will be ineffective model which will also be affecting the computational cost and energy required.

Therefore, in the future research, we will realize cognitive locomotion which generates multiple gaits depending on the 3 aspects, embodiment, locomotion generator, and cognition model. We will design the dynamic neuro-locomotion integrated with internal and external sensory information for correlating with the environmental condition, where its illustration can be seen in Figure 8.1.

# References


[1] Basic motion planning model. https://en.wikipedia.org/wiki/Motion_planning. Accessed: 2018-01-30.

[2] Bigdog. https://www.bostondynamics.com/bigdog. Accessed: 2017-12-19.

[3] D-h parameters. https://en.wikipedia.org/wiki/Denavit\OT1\textendashHartenberg_parameters. Accessed: 2017-12-20.

[4] iPhone 5. http://www.apple.com/iphone-5c/specs/. Accessed: 2018-01-30.

[5] Locomotion concept. . https://www.ethz.ch/content/dam/ethz/special-interest/mavt/robotics-n-intelligent-systems/asl-dam/documents/lectures/autonomous_mobile_robots/spring-2017/Locomotion%20Concepts.pdf. Accessed: 2017-12-22.

[6] Robotis comprehensive kit. http://support.robotis.com/en/product/bioloid/bioloid_comp_main.htm. Accessed: 2018-01-30.

[7] Robotis dynamixel ax-12. http://support.robotis.com/en/product/dynamixel/ax_series/dxl_ax_actuator.htm. Accessed: 2018-01-30.

[8] WABOT -WAseda roBOT-. http://www.humanoid.waseda.ac.jp/booklet/kato_2.html. Accessed: 2017-12-19.

[9] Muhammad Abdallah and Ambarish Goswami. A biomechanically motivated two-phase strategy for biped upright balance control. In *Proc. of ICRA*, pages 1996–2001, 2005.

[10] Zofia Afelt and Stefal Kasicki. Limb coordination during locomotion in cats and dogs. *ACTA Neurobiological*, 35:369–378, 1975.

[11] Mohiuddin Ahmed, MR Khan, Masum Billah, and Soheli Farhana. *Walking hexapod robot in disaster recovery: developing algorithm for terrain negotiation and navigation.* INTECH Open Access Publisher, 2010.

[12] Juan José Alcaraz-Jimenez, David Herrero-Perez, and H Martınez-Barberá. Motion planning for omnidirectional dynamic gait in humanoid soccer robots. *Journal of Physical Agents*, 5(1):25–34, 2011.

[13] Ahmed RJ Almusawi, L Canan Dülger, and Sadettin Kapucu. A new artificial neural network approach in solving inverse kinematics of robotic arm (denso vp6242). *Computational intelligence and neuroscience*, 2016, 2016.

[14] Elmira Amrollah and Patrick Henaff. On the role of sensory feedbacks in rowat–selverston cpg to improve robot legged locomotion. *Frontiers in neurorobotics*, 4, 2010.

[15] J. A. Anderson and E. Rosenfeld. *Neurocomputing*. MIT Press, 1988.

[16] Farooq Azam. *Biologically Inspired Modular Neural Networks*. PhD thesis, Virginia Polytechnic Institute and State University, Blacksburg, Virginia, May 2000.

[17] Justin Jeffrey Baker. Dexterous finger movements: Decoding neuro- and myoelectric signals and properties of finger-related neurons in motor cortex. Master's thesis, The University of Utah, 12 2010.

[18] K. Balázs, J. Botzheim, and L. T. Kóczy. Comparative investigation of various evolutionary and memetic algorithms. *Computational Intelligence in Engineering, ser. Studies in Computational Intelligence*, 313:129–140, March 2010.

[19] A Balestrino, G De Maria, and L Sciavicco. Robust control of robotic manipulators. *IFAC Proceedings Volumes*, 17(2):2435–2440, 1984.

[20] Rudi Ball and Naranker Dulay. Enhancing traffic intersection control with intelligent objects. *Urban Internet of Things Towards Programmable Realtime Cities*, 2010.

[21] Jacky Baltes, Sara McGrath, and John Anderson. Active balancing using gyroscopes for a small humanoid robot. In *2nd International Conference on Autonomous Robots and Agents*, pages 13–15. Citeseer, 2004.

[22] Atilim Gunes Baydin. Evolution of central pattern generators for the control of a five-link bipedal walking mechanism. *Paladyn journal of behavioral robotics*, 3(1):45–53, 2012.

[23] Bradley E Bishop, Frederick L Crabbe, and Bryan M Hudock. Design of a low-cost, highly mobile urban search and rescue robot. *Advanced Robotics*, 19(8):879–899, 2005.

[24] Geoff Boeing. Visual analysis of nonlinear dynamical systems: Chaos, fractals, self-similarity and the limits of prediction. *Systems*, 4(4):37, 2016.

[25] J. Botzheim, C. Cabrita, L. T. Kóczy, and A. E. Ruano. Fuzzy rule extraction by bacterial memetic algorithms. *International Journal of Intelligent Systems*, 24(3):312–339, March 2009.

[26] János Botzheim, Yuichiro Toda, and Naoyuki Kubota. Bacterial memetic algorithm for offline path planning of mobile robots. *Memetic Comp.*, 4(1):73–86, 2012.

[27] Alexandros Bouganis and Murray Shanahan. Training a spiking neural network to control a 4-dof robotic arm based on spike timing-dependent plasticity. In *Neural Networks (IJCNN), The 2010 International Joint Conference on*, pages 1–8. IEEE, 2010.

[28] Martim Brandao, Kenji Hashimoto, José Santos-Victor, and Atsuo Takanishi. Footstep planning for slippery and slanted terrain using human-inspired models. *IEEE Transactions on Robotics*, 32(4):868–879, 2016.

[29] Gal Chechik, Isaac Meilijson, and Eytan Ruppin. Synaptic pruning in development: a computational account. *Neural Comp.*, 10(7):1759–1777, 1998.

[30] Gal Chechik, Isaac Meilijson, and Eytan Ruppin. Neuronal regulation: A mechanism for synaptic pruning during brain maturation. *Neural Comp.*, 11(8):2061–2080, 1999.

[31] S. Chernova and M. Veloso. An evolutionary approach to gait learning for four-legged robots. In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, volume 3, pages 2562–2567, Sept 2004.

[32] Joel Chestnutt, Manfred Lau, German Cheung, James Kuffner, Jessica Hodgins, and Takeo Kanade. Footstep planning for the honda asimo humanoid. In *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*, pages 629–634. IEEE, 2005.

[33] C. Chevallereau and Y. Aoustin. Self-stabilization of 3D walking via vertical oscillations of the hip. In *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 5088–5093, May 2015.

[34] Christine Chevallereau, Gabriel Abba, Yannick Aoustin, Franck Plestan, Eric Westervelt, Carlos Canudas de Wit, and Jessy Grizzle. Rabbit: A testbed for advanced control theory. *IEEE Control Systems Magazine*, 23(5):57–79, 2003.

[35] Holk Cruse. *Neural Networks as Cybernetic Systems - 2rd and revised edition*. Brains, Minds & Media, Bielefeld, Germany, October 2006.

[36] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, 2002.

[37] Robin Deits and Russ Tedrake. Footstep planning on uneven terrain with mixed-integer convex optimization. In *Humanoid Robots (Humanoids), 2014 14th IEEE-RAS International Conference on*, pages 279–286. IEEE, 2014.

[38] Nicolas Delanoue, Luc Jaulin, and Bertrand Cottenceau. Counting the number of connected components of a set and its application to robotics. In *International Workshop on Applied Parallel Computing*, pages 93–101. Springer, 2004.

[39] J. Denavit and R. S. Hartenberg. A kinematic notation for lower-pair mechanisms based on matrices. *Trans. ASME, J. Appl. Mech.*, 22(2):215 – 221, 1965.

[40] Jacques Denavit. A kinematic notation for lower-pair mechanisms based on matrices. *ASME J. Appl. Mech.*, pages 215–221, 1955.

[41] Arati S Deo and Ian D Walker. Adaptive non-linear least squares for inverse kinematics. In *Robotics and Automation, 1993. Proceedings., 1993 IEEE International Conference on*, pages 186–193. IEEE, 1993.

[42] Edsger W Dijkstra. A note on two problems in connexion with graphs. *Numerische mathematik*, 1(1):269–271, 1959.

[43] S. Dogru and L. Marques. Energy efficient coverage path planning for autonomous mobile robots on 3D terrain. In *Proc. of IEEE Int. Conf. on Auton. Robot Syst. and Competitions*, pages 118–123, April 2015.

[44] S. Dogru and L. Marques. Towards fully autonomous energy efficient coverage path planning for autonomous mobile robots on 3D terrain. In *Proc. of European Conf. on Mobile Robots*, pages 1–6, Sept 2015.

[45] Masahiro Doi, Y Hmegawa, and Toshio Fukuda. Passive dynamic autonomous control of bipedal walking. In *Humanoid Robots, 2004 4th IEEE/RAS International Conference on*, volume 2, pages 811–829. IEEE, 2004.

[46] Aaron D'Souza, Sethu Vijayakumar, and Stefan Schaal. Learning inverse kinematics. In *Intelligent Robots and Systems, 2001. Proceedings. 2001 IEEE/RSJ International Conference on*, volume 1, pages 298–303. IEEE, 2001.

[47] Shival Dubey, Manish Prateek, and Mukesh Saxena. Robot locomotion-a review. *International Journal of Applied Engineering Research*, 10(3):7357–69, 2015.

[48] Arduino Due. Arduino. *Saatavissa: http://arduino. cc/en/Main/arduinoBoardDue. Hakupäivä*, 4, 2014.

[49] Adrian-Vasile Duka. Neural network based inverse kinematics solution for trajectory tracking of a robotic arm. *Procedia Technology*, 12:20–27, 2014.

[50] Adrian-Vasile Duka. Anfis based solution to the inverse kinematics of a 3dof planar manipulator. *Procedia Technology*, 19:526–533, 2015.

[51] J. Duysens and HW. Van de Crommert. Neural control of locomotion; the central pattern generator from cats to humans. *Gait Posture*, 7(2):131–141, Mar 1998.

[52] Gen Endo, Jun Morimoto, Jun Nakanishi, and Gordon Cheng. An empirical exploration of a neural oscillator for biped locomotion control. In *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on*, volume 3, pages 3036–3042. IEEE.

[53] Dave Ferguson and Anthony Stentz. Field D*: An interpolation-based path planner and replanner. In *Robot. Res.*, pages 239–253. 2007.

[54] J Ferreira, Manuel Crisóstomo, and A Coimbra. Neuro-fuzzy zmp control of a biped robot. In *Proceedings of the 6th WSEAS International Conference on Simulation, Modeling and Optimization*, pages 331–337, 2006.

[55] Jerome H Friedman. Multivariate adaptive regression splines. *The annals of statistics*, pages 1–67, 1991.

[56] B. Fritzke. Growing cell structures – a self-organizing network in $k$ dimensions. *Artificial Neural Networks*, 2(2):1051–1056, 1994.

[57] B. Fritzke. A growing neural gas network learns topologies. *Advances in Neural Information Processing Systems*, 7:625–632, 1995.

[58] B. Fritzke. Growing self-organizing networks – why? In *Proc. of the European Symposium on Artificial Neural Networks*, pages 61–72, 1996.

[59] T. Fukuda, Y. Komata, and T. Arakawa. Recurrent neural network with self-adaptive gas for biped locomotion robot. In *Neural Networks,1997., International Conference on*, volume 3, pages 1710–1715 vol.3, Jun 1997.

[60] T. Fukuda, Y. Komata, and T. Arakawa. Stabilization control of biped locomotion robot based learning with gas having self-adaptive mutation and recurrent neural networks. In *Robotics and Automation, 1997. Proceedings., 1997 IEEE International Conference on*, volume 1, pages 217–222 vol.1, Apr 1997.

[61] Toshio Fukuda, Yasuhisa Hasegawa, Kosuke Sekiyama, and Tadayoshi Aoyama. *Multi-Locomotion Robotic Systems: New Concepts of Bio-inspired Robotics*, volume 81. Springer, 2012.

[62] S. Gay, J. Santos-Victor, and A. Ijspeert. Learning robot gait stability using neural networks as sensory feedback function for central pattern generators. In *Proc. of IROS*, pages 194–201, Nov 2013.

[63] W. Gerstner, W. Maass, and C. M. Bishop. *Pulsed Neurol Networks*. MIT Press, 1999.

[64] Roy Glasius, Andrzej Komoda, and Stan CAM Gielen. Neural network dynamics for path planning and obstacle avoidance. *Neural Netw.*, 8(1):125–133, 1995.

[65] Jessy W Grizzle, Gabriel Abba, and Franck Plestan. Asymptotically stable walking for biped robots: Analysis via systems with impulse effects. *IEEE Transactions on automatic control*, 46(1):51–64, 2001.

[66] Stephen Grossberg. Nonlinear neural networks: Principles, mechanisms, and architectures. *Neural Netw.*, 1(1):17 – 61, 1988.

[67] Xiaodong Gu, Liming Zhang, and Daoheng Yu. Delay PCNN and its application for optimization. In *Proc. of Int. Symp. on Neural Netw.*, pages 413–418, 2004.

[68] Tuomas Haarnoja. Dynamic modeling and velocity control for limit cycle walking, 2010.

[69] Richard Scheunemann Hartenberg and Jacques Denavit. *Kinematic synthesis of linkages*. McGraw-Hill, 1964.

[70] Kenji Hashimoto, Shunsuke Kimura, Nobuaki Sakai, Shinya Hamamoto, Ayanori Koizumi, Xiao Sun, Takashi Matsuzawa, Tomotaka Teramachi, Yuki Yoshida, Asaki Imai, et al. Warec-1-a four-limbed robot having high locomotion ability with versatility in locomotion styles.

[71] T. Hastie, R. Tibshirani, and J. Friedman. *The elements of statistical learning; Data mining, Inference, and Prediction*. Springer Verlag, 2001.

[72] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*. Springer Series in Statistics. Springer New York Inc., New York, NY, USA, 2001.

[73] Michiel Hazewinkel. Bezier curve. *Encyclopedia of Mathematics*, 2001.

[74] Peter Henry, Michael Krainin, Evan Herbst, Xiaofeng Ren, and Dieter Fox. RGB-D mapping: Using kinect-style depth cameras for dense 3D modeling of indoor environments. *The Int. Journal of Robot. Res.*, 31(5):647–663, 2012.

[75] Carlos Hernández and Jorge A Baier. Making A* run faster than D*-lite for path-planning in partially known terrain. 2014.

[76] S. Hirose, S. Yokota, A. Torii, M. Ogata, S. Suganuma, K. Takita, and K. Kato. Quadruped walking robot centered demining system - development of titan-ix and its operation-. In *IEEE International Conference on Robotics and Automation*, pages 1284–1290, 2005.

[77] Daan GE Hobbelen and Martijn Wisse. Controlling the walking speed in limit cycle walking. *The International Journal of Robotics Research*, 27(9):989–1005, 2008.

[78] Alan L Hodgkin and Andrew F Huxley. A quantitative description of membrane current and its application to conduction and excitation in nerve. *The Journal of Physiology*, 117(4):500, 1952.

[79] Young-Dae Hong, Chang-Soo Park, and Jong-Hwan Kim. Stable bipedal walking with a vertical center-of-mass motion by an evolutionary optimized central pattern generator. *IEEE Transactions on Industrial Electronics*, 61(5):2346–2355, May 2014.

[80] Peter R Huttenlocher. Synaptic density in human frontal cortex—developmental changes and effects of aging. *Brain Res.*, 163(2):195–205, 1979.

[81] Sang-Ho Hyon and Gordon Cheng. Disturbance rejection for biped humanoids. In *Proc. of ICRA*, pages 2668–2675, 2007.

[82] Sang-Ho Hyon, Rieko Osu, and Yohei Otaka. Integration of multi-level postural balancing on humanoid robots. In *Proc. of ICRA*, pages 1549–1556, 2009.

[83] A.J. Ijspeert. A connectionist central pattern generator for the aquatic and terrestrial gaits of a simulated salamander. *Biological Cybernetics*, 85(5):331–348, 2001.

[84] Auke Jan Ijspeert. Central pattern generators for locomotion control in animals and robots: a review, 2008.

[85] Auke Jan Ijspeert, Alessandro Crespi, and Jean-Marie Cabelguen. Simulation and robotics studies of salamander locomotion - applying neurobiological principles to the control of locomotion in robots. *Neuroinformatics*, 3(3):171–195, 2005.

[86] AukeJan Ijspeert and Jean-Marie Cabelguen. Gait transition from swimming to walking: Investigation of salamander locomotion control using nonlinear oscillators. In Hiroshi Kimura, Kazuo Tsuchiya, Akio Ishiguro, and Hartmut Witte, editors, *Adaptive Motion of Animals and Machines*, pages 177–188. Springer Tokyo, 2006.

[87] H. Inada and K. Ishii. Behavior generation of bipedal robot using central pattern generator(CPG) (1st report: CPG parameters searching method by genetic algorithm). In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, volume 3, pages 2179–2184, Oct 2003.

[88] Akio Ishiguro, Akinobu Fujii, and Peter Eggenberger Hotz. Neuromodulated control of bipedal locomotion using a polymorphic CPG circuit. *Adaptive Behaviour*, 11(1):7–17, 2003.

[89] T. Iwasaki. Biological control mechanisms for oscillation and locomotion. In *Proc. of the 31st Chinese Control Conference (CCC)*, pages 27–30, July 2012.

[90] Mikkel T. Jensen. Reducing the run-time complexity of multiobjective eas: The NSGA-II and other algorithms. *IEEE Trans. Evolutionary Computation*, 7(5):503–515, 2003.

[91] Shuuji Kajita and Bernard Espiau. Legged robots. In *Springer handbook of robotics*, pages 361–389. Springer, 2008.

[92] Shuuji Kajita, Fumio Kanehiro, Kenji Kaneko, Kiyoshi Fujiwara, Kensuke Harada, Kazuhito Yokoi, and Hirohisa Hirukawa. Biped walking pattern generation by using preview control of zero-moment point. In *Robotics and Automation, 2003. Proceedings. ICRA'03. IEEE International Conference on*, volume 2, pages 1620–1626. IEEE, 2003.

[93] Shuuji Kajita, Fumio Kanehiro, Kenji Kaneko, Kazuhito Yokoi, and Hirohisa Hirukawa. The 3d linear inverted pendulum mode: A simple modeling for a biped walking pattern generation. In *Intelligent Robots and Systems, 2001. Proceedings. 2001 IEEE/RSJ International Conference on*, volume 1, pages 239–246. IEEE, 2001.

[94] Dimitrios Alexios Karras. An improved modular neural network model for adaptive trajectory tracking control of robot manipulators. 5506:1063–1070, 2009.

[95] SH Kasaei, SM Kasaei, and SA Kasaei. A hybrid neural network controller for stable walking of a humanoid soccer robot. *Annals. Computer Science Series*, 8(1), 2010.

[96] Jung-Yup Kim, Ill-Woo Park, and Jun-Ho Oh. Experimental realization of dynamic walking of the biped humanoid robot KHR-2 using zero moment point feedback and inertial measurement. *Advanced Robotics*, 20(6):707–736, 2006.

[97] Jung-Yup Kim, Ill-Woo Park, and Jun-Ho Oh. Walking control algorithm of biped humanoid robot on uneven and inclined floor. *Journal of Intelligent & Robotic Systems*, 48(4):457–484, 2007.

[98] Ryan Kiros, Ruslan Salakhutdinov, and Rich Zemel. Multimodal neural language models. In *31st Intl. Conf. on Machine Learning*, pages 595–603, 2014.

[99] T. Kohonen. *Self-Organizing Maps*. Springer, Berlin, 2001.

[100] RaşIt KöKer. A genetic algorithm approach to a neural-network-based inverse kinematics solution of robotic manipulators based on error minimization. *Information Sciences*, 222:528–543, 2013.

[101] Raşit Köker, Cemil Öz, Tarık Çakar, and Hüseyin Ekiz. A study of neural network based inverse kinematics solution for a three-joint robot. *Robotics and autonomous systems*, 49(3):227–234, 2004.

[102] Jung-Shik Kong, Eung-Hyuk Lee, Bo-Hee Lee, and Jin-Geol Kim. Study on the real-time walking control of a humanoid robot using fuzzy algorithm. *International Journal of Control, Automation, and Systems*, 6(4):551–558, 2008.

[103] V. Kroumov and Jianli Yu. 3D path planning for mobile robots using annealing neural network. In *Proc. of Int. Conf. on Networking, Sensing and Control*, pages 130–135, March 2009.

[104] N. Kubota and T. Fukuda. *6.43.38.4 Intelligent Robots*. in Control Systems, Robotics and Automation, edited by H. Unbehauen, in Encyclopedia of Life Support Systems (EOLSS), Developed under the auspices of the UNESCO, Oxford, UK.

[105] Scott Kuindersma, Robin Deits, Maurice Fallon, Andrés Valenzuela, Hongkai Dai, Frank Permenter, Twan Koolen, Pat Marion, and Russ Tedrake. Optimization-based locomotion planning, estimation, and control design for the atlas humanoid robot. *Autonomous Robots*, 40(3):429–455, 2016.

[106] Scott Kuindersma, Frank Permenter, and Russ Tedrake. An efficiently solvable quadratic program for stabilizing dynamic locomotion. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pages 2589–2594. IEEE, 2014.

[107] H.M. Lakany, G.M. Hayes, M.E. Hazlewood, and S.J. Hillman. Human walking: tracking and analysis. In *Motion Analysis and Tracking (Ref. No. 1999/103), IEE Colloquium on*, pages 5/1–5/14, 1999.

[108] Cecilia Laschi, Paolo Dario, Maria Chiara Carrozza, Eugenio Guglielmelli, Giancarlo Teti, Davide Taddeucci, Fabio Leoni, Bruno Massa, Massimiliano Zecca, and Roberto Lazzarini. Grasping and manipulation in humanoid robotics. In *IEEE International Conference on Humanoid Robots (Humanoids 2003)*, 2000.

[109] Jean-Claude Latombe. *Robot motion planning*, volume 124. Springer Science & Business Media, 2012.

[110] H. Li and Q. Zhang. Multiobjective optimization problems with complicated Pareto sets, MOEA/D and NSGA-II. *IEEE Trans. on Evolutionary Computation*, 13(2):284–302, 2008.

[111] Xiaojun Li, Yide Ma, and Xiaowen Feng. Self-adaptive autowave pulse-coupled neural network for shortest-path problem. *Neurocomp.*, 115:63–71, 2013.

[112] Yan Li and Yong Wang. Trajectory tracking control of a redundantly actuated parallel robot using diagonal recurrent neural network. In *5th Intl. Conf. on Natural Computation*, volume 2, pages 292–296, Aug 2009.

[113] Maxim Likhachev, David I Ferguson, Geoffrey J Gordon, Anthony Stentz, and Sebastian Thrun. Anytime dynamic A*: An anytime, replanning algorithm. 2005.

[114] Maxim Likhachev, Geoffrey J Gordon, and Sebastian Thrun. ARA*: Anytime A* with provable bounds on sub-optimality. In *Adv. in Neural Inform. Process. Syst.*, 2003.

[115] C. Lin and E. Boldbaatar. Fault accommodation control for a biped robot using a recurrent wavelet Elman neural network. *Systems Journal, IEEE*, PP(99):1–12, 2015.

[116] F.-J. Lin, P.-H. Shieh, and P.-H. Shen. Robust recurrent-neural-network sliding-mode control for the x-y table of a cnc machine. *IEEE Proceedings Control Theory and Applications*, 153(1):111–123, Jan 2006.

[117] Chengju Liu, Qijun Chen, and Danwei Wang. CPG-inspired workspace trajectory generation and adaptive locomotion control for quadruped robots. *IEEE Transaction on System, Man, and Cybernetics*, 41(5):867–880, September 2011.

[118] Chengju Liu, Danwei Wang, and Qijun Chen. Central pattern generator inspired control for adaptive walking of biped robots. *IEEE Transaction on System, Man, and Cybernetics*, 43:1206–1215, September 2013.

[119] Guisong Liu, Zhao Qiu, Hong Qu, and Luping Ji. Computing k shortest paths using modified pulse-coupled neural network. *Neurocomp.*, 149:1162–1176, 2015.

[120] Jinsu Liu and Manuela Veloso. Online zmp sampling search for biped walking planning. In *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*, pages 185–190. IEEE, 2008.

[121] Ren C Luo and Chin Cheng Chen. Biped walking trajectory generator based on three-mass with angular momentum model using model predictive control. *IEEE Transactions on Industrial Electronics*, 63(1):268–276, 2016.

[122] Wolfgang Maass. Computing with spiking neurons. In *Pulsed neural networks*, pages 55–85. MIT Press, 1999.

[123] E. H. Mamdani and S. Assilian. An experiment in linguistic synthesis with a fuzzy logic controller. *Int. Journal Man-Mach Stud*, 7:1–13, 1975.

[124] Poramate Manoonpong, Frank Pasemann, and Florentin Wörgötter. Sensor-driven neural control for omnidirectional locomotion and versatile reactive behaviors of walking machines. *Robotics and Autonomous Systems*, 56(3):265–288, 2008.

[125] T. M. Martinetz and K. J. Schulten. A 'neural gas' network learns topologies. *Artificial Neural Networks*, 1:397–402, 1991.

[126] L.S. Martins-Filho and R. Prajoux. Locomotion control of a four-legged robot embedding real-time reasoning in the force distribution. *Robotics and Auton. Syst.*, 32(4):219 – 235, 2000.

[127] Vítor Matos and Cristina P. Santos. Towards goal-directed biped locomotion: Combining CPGs and motion primitives. *Robotics and Autonomous Systems*, 62(12):1669–1690, 2014.

[128] K. Matsuoka. Mechanisms of frequency and pattern control in the neural rhythm generators. *Biological Cybernetics*, 56:345–353, 1987.

[129] K. Matsuoka. Mechanisms of frequency and pattern control in the neural rhythm generators. *Biological Cybernetics*, 56:345–353, 1987.

[130] Kiyotoshi Matsuoka. Sustained oscillations generated by mutually inhibiting neurons with adaptation. *Biological Cybernetics*, 52:367–376, 1985.

[131] Kiyotoshi Matsuoka. Analysis of a neural oscillator. *Biological cybernetics*, 104(4):297–304, 2011.

[132] Takashi Matsuzawa, Ayanori Koizumi, Kenji Hashimoto, Xiao Sun, Shinya Hamamoto, Tomotaka Teramachi, Nobuaki Sakai, Shunsuke Kimura, and Atsuo Takanishi. Crawling motion and foot trajectory modification control for legged robot on rough terrain. In *Mechatronics and Automation (ICMA), 2017 IEEE International Conference on*, pages 1976–1982. IEEE, 2017.

[133] Christophe Maufroy, Hiroshi Kimura, and Kunikatsu Takase. Towards a general neural controller for quadrupedal locomotion. *Neural Networks*, 21(4):667–681, 2008.

[134] Tad McGeer et al. Passive dynamic walking. *I. J. Robotic Res.*, 9(2):62–82, 1990.

[135] Robert B McGhee and Andrew A Frank. On the stability properties of quadruped creeping gaits. *Mathematical Biosciences*, 3:331–351, 1968.

[136] S. Miyakoshi, G. Taga, Y. Kuniyoshi, and A. Nagakubo. Three dimensional bipedal stepping motion using neural oscillators-towards humanoid motion in the real world. In *Intelligent Robots and Systems, 1998. Proceedings., 1998 IEEE/RSJ International Conference on*, volume 1, pages 84–89 vol.1, Oct 1998.

[137] Néstor Morales, Jonay Toledo, and Leopoldo Acosta. Path planning using a multiclass support vector machine. *Applied Soft Comp.*, 43:498–509, 2016.

[138] Zahra Moravej, Farhad Adelnia, and Fazel Abbasi. Optimal coordination of directional overcurrent relays using NSGA-II. *Electric Power Systems Research*, 119:228–239, 2015.

[139] Yoshikazu Mori, Kazuhiro Takayama, Takeshi Adachi, Shintaro Omote, and Tatsuya Nakamura. Feasibility study on an excavation-type demining robot. *Auton. Robots*, 18(3):263–274, 2005.

[140] Yoshihiko Nakamura and Hideo Hanafusa. Inverse kinematic solutions with singularity robustness for robot manipulator control. *ASME, Transactions, Journal of Dynamic Systems, Measurement, and Control*, 108:163–171, 1986.

[141] J. Nassour, P. Henaff, F. Benouezdou, and G. Cheng. Experience-based learning mechanism for neural controller adaptation: Application to walking biped robots. In *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, pages 2616–2621, Oct 2009.

[142] John Nassour, Patrick Henaff, Fethi Ben Ouezdou, and Gordon Cheng. Multi-layered multi-pattern CPG for adaptive locomotion of humanoid robots. *Biological Cybernetics*, pages 291–303, 2014.

[143] Nils J Nilsson. *Principles of artificial intelligence*. Springer-Verlag Berlin Heidelberg, 2014.

[144] N.M. Nor and S. Ma. CPG-based locomotion control of a snake-like robot for obstacle avoidance. In *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 347–352, May 2014.

[145] D. Ortiz-Arroyo. A hybrid 3D path planning method for UAVs. In *Proc. of Workshop on Res., Edu. and Develop. of Unmanned Aerial Systems*, pages 123–132, Nov 2015.

[146] Dai Owaki, Leona Morikawa, and Akio Ishiguro. Gait transition of quadruped robot without interlimb neural connections. In *Proc. of dynamic walking*, 2012.

[147] Eimei Oyama, Nak Young Chong, Arvin Agah, and Taro Maeda. Inverse kinematics learning by modular architecture neural networks with performance prediction networks. In *Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on*, volume 1, pages 1006–1012. IEEE, 2001.

[148] Chang-Soo Park, Young-Dae Hong, and Jong-Hwan Kim' Full-body joint trajectory generation using an evolutionary central pattern generator for stable bipedal walking. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pages 160–165, Oct 2010.

[149] Chang-Soo Park, Young-Dae Hong, and Jong-Hwan Kim. Evolutionary-optimized central pattern generator for stable modifiable bipedal walking. *IEEE/ASME Transactions on Mechatronics*, 19(4):1374–1383, Aug 2014.

[150] Richard P Paul. *Robot manipulators: mathematics, programming, and control: the computer control of robot manipulators*. Richard Paul, 1981.

[151] Rolf Pfeifer and Josh Bongard. *How the body shapes the way we think: a new view of intelligence*. MIT press, 2006.

[152] Hong Qu, Simon X Yang, Allan R Willms, and Zhang Yi. Real-time robot path planning based on a modified pulse-coupled neural network model. *IEEE Trans. on Neural Netw.*, 20(11):1724–1739, 2009.

[153] Hong Qu, Zhang Yi, and Simon X Yang. Efficient shortest-path-tree computation in network routing based on pulse-coupled neural networks. *IEEE Trans. on Cybern.*, 43(3):995–1010, 2013.

[154] Mathias Quoy, Sorin Moga, and Philippe Gaussier. Dynamical neural networks for planning and low-level robot control. *IEEE Trans. on Syst., Man and Cybern., Part A: Syst. and Humans*, 33(4):523–532, 2003.

[155] Marc Raibert, Kevin Blankespoor, Gabriel Nelson, and Rob Playter. Bigdog, the rough-terrain quadruped robot. *IFAC Proceedings Volumes*, 41(2):10822–10825, 2008.

[156] Amar Ramdane-Cherif, Boubaker Daachi, Abdelaziz Benallegue, and Nicole Lévy. Kinematic inversion. In *Intelligent Robots and Systems, 2002. IEEE/RSJ International Conference on*, volume 2, pages 1904–1909. IEEE, 2002.

[157] T. Reil and P. Husbands. Evolution of central pattern generators for bipedal walking in a real-time physics environment. *IEEE Transactions on Evolutionary Computation*, 6(2):159–168, Apr 2002.

[158] Guanjiao Rena, Weihai Chena, Sakyasingha Dasguptab, Florentin Wrgtterb Christoph Kolodziejskib, and Poramate Manoonpong. Multiple chaotic central pattern

generators with learning for legged locomotion and malfunction compensation. *Information Sciences*, 294:666–682, 2015.

[159] J. Rizo, J. Coronado, A. Forero, C. Otalora, and M. Devy. URSULA: robotic demining system. In *Proc. of the 11th Int. Conf. on Adv. Robot.*, pages 538–543, 2003.

[160] Ulrich Roth, Marc Walker, Arne Hilmann, and Heinrich Klar. Dynamic path planning with spiking neural networks. In *Biological and Artificial Computation: From Neuroscience to Technology*, pages 1355–1363. 1997.

[161] Peter F. Rowat, Allen, and I. Selverston. Oscillatory mechanisms in pairs of neurons connected with fast inhibitory synapses. *J. Comp. Neurosci*, pages 103–127, 1997.

[162] Peter F Rowat and Allen I Selverston. Learning algorithms for oscillatory networks with gap junctions and membrane currents. *Network*, pages 17–41, 1991.

[163] Peter F Rowat and Allen I Selverston. Modeling the gastric mill central pattern generator of the lobster with a relaxation-oscillator network. *Journal of neurophysiology*, 70(3):1030–1053, 1993.

[164] A. Roy. Connectionism, controllers, and a brain theory. *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, 38(6):1434–1441, Nov 2008.

[165] S. Rutishauser, A. Sprowitz, L. Righetti, and A. J. Ijspeert. Passive compliant quadruped robot using central pattern generators for locomotion control. In *Proc. of 2nd IEEE RAS EMBS Intl. Conf. on Biomed. Robot. Biomechatron.*, pages 710–715, Oct 2008.

[166] L. R. Sahawneh, M. E. Argyle, and R. W. Beard. 3D path planning for small UAS operating in low-altitude airspace. In *Proc. of Int. Conf. on Unmanned Aircraft Systems*, pages 413–419, June 2016.

[167] Y. Sakagami, R. Watanabe, C. Aoyama, S. Matsunaga, N. Higaki, and K. Fujimura. The intelligent ASIMO: system overview and integration. In *Intelligent Robots and Systems, 2002. IEEE/RSJ International Conference on*, volume 3, pages 2478–2483 vol.3, 2002.

[168] A. A. Saputra, J. Botzheim, I. A. Sulistijono, and N. Kubota. Biologically inspired control system for 3-d locomotion of a humanoid biped robot. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 46(7):898–911, July 2016.

[169] A. A. Saputra, A. S. Khalilullah, I. A. Sulistijono, and N. Kubota. Adaptive motion pattern generation on balancing of humanoid robot movement. In *Proc. of CCECE*, pages 1479–1484, May 2015.

[170] Azhar Aulia Saputra, János Botzheim, and Naoyuki Kubota. Movement control on neural oscillator based humanoid robot locomotion. In *Proc. of 21st Intelligent Mechatronics Workshop*, pages 126–131, Hakodate, Japan, August 2016.

[171] Azhar Aulia Saputra, János Botzheim, and Naoyuki Kubota. Walking speed control in human behavior inspired gait generation system for biped robot. In *Evolutionary Computation (CEC), 2016 IEEE Congress on*, pages 4895–4902. IEEE, 2016.

[172] Azhar Aulia Saputra, Achmad Subhan Khalilullah, and Naoyuki Kubota. *Development of Humanoid Robot Locomotion Based on Biological Approach in EEPIS Robot Soccer (EROS)*. Springer International Publishing, 2015.

[173] Azhar Aulia Saputra, Achmad Subhan Khalilullah, and Indra Adji Sulistijono. Acceleration and deceleration optimization using inverted pendulum model on humanoid robot EROS-2. In *Proc. of the Indonesian Symposium on Robot Soccer Competition*, pages 17–22, Yogyakarta, Indonesia, July 2014.

[174] Azhar Aulia Saputra, Achmad Subhan Khalilullah, Indra Adji Sulistijono, and Naoyuki Kubota. Adaptive motion pattern generation on balancing of humanoid robot movement. In *in Proc. of 28th Canadian Conf. on Electr. and Comp. Eng.*, pages 1479–1484, 2015.

[175] Azhar Aulia Saputra, Indra Adji Sulistijono, J. Botzheim, and Naoyuki Kubota. Interconnection structure optimization for neural oscillator based biped robot locomotion. In *Proc. of the IEEE Symposium on Robotic Intelligence in Informationally Structured Space*, 2015, Accepted.

[176] Azhar Aulia Saputra, Indra Adji Sulistijono, Achmad Subhan Khalilullah, Takahiro Takeda, and Naoyuki Kubota. Combining pose control and angular velocity control for motion balance of humanoid robot soccer EROS. In *Proc. of the IEEE Symposium on Robotic Intelligence in Informatically Structured Space*, pages 16–21, Orlando, USA, December 2014.

[177] Azhar Aulia Saputra, T. Takeda, and N. Kubota. Efficiency energy on humanoid robot walking using evolutionary algorithm. In *Proc. of the IEEE Congress on Evolutionary Computation (CEC)*, pages 573–578, Sendai, Japan, May 2015.

[178] Azhar Aulia Saputra, Takahiro Takeda, János Botzheim, and Naoyuki Kubota. Multi-objective evolutionary algorithm for neural oscillator based robot locomotion. In *Proc. of the 41st IEEE Ind. Elec. Soc.*, pages 2655–2660, 2015.

[179] C. Saranya, Manju Unnikrishnan, S. Akbar Ali, D.S. Sheela, and Dr. V.R. Lalithambika. Terrain based d algorithm for path planning. *IFAC-PapersOnLine*, 49(1):178 – 182, 2016.

[180] E.L. Secco and G. Magenes. A feedforward neural network controlling the movement of a 3-DOF finger. *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, 32(3):437–445, May 2002.

[181] Claudio Semini, Nikos G Tsagarakis, Emanuele Guglielmino, Michele Focchi, Ferdinando Cannella, and Darwin G Caldwell. Design of hyq–a hydraulically and electrically actuated quadruped robot. *Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering*, 225(6):831–849, 2011.

[182] Lukas Silberbauer. A new 6 wheeled robot for humanitarian demining. In *EURON/IARP Int. Workshop on Robot. for Risky Interventions and Surveillance of the Environment*, 2008.

[183] Russell Smith et al. Open dynamics engine. 2005.

[184] Seungmoon Song. *Development of an omni-directional gait generator and a stabilization feedback controller for humanoid robots*. PhD thesis, Virginia Polytechnic Institute and State University, 2010.

[185] N. Srinivas and Kalyanmoy Deb. Multiobjective optimization using nondominated sorting in genetic algorithms. *Evolutionary Computation*, 2:221–248, 1994.

[186] Benjamin Stephens. Humanoid push recovery. In *in International Conference on Humanoid Robots*, pages 589–595, 2007.

[187] Johannes Strom, George Slavov, and Eric Chown. Omnidirectional walking using zmp and preview control for the nao humanoid robot. In *Robot Soccer World Cup*, pages 378–389. Springer, 2009.

[188] Shigeki Sugano and Ichiro Kato. Wabot-2: Autonomous robot with dexterous finger-arm–finger-arm coordination control in keyboard performance. In *Robotics and Automation. Proceedings. 1987 IEEE International Conference on*, volume 4, pages 90–97. IEEE, 1987.

[189] Tomomichi Sugihara. Simulated regulator to synthesize zmp manipulation and foot location for autonomous control of biped robots. In *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, pages 1264–1269. IEEE, 2008.

[190] Tomomichi Sugihara, Yoshihiko Nakamura, and Hirochika Inoue. Real-time humanoid motion generation through zmp manipulation based on inverted pendulum control. In *Robotics and Automation, 2002. Proceedings. ICRA'02. IEEE International Conference on*, volume 2, pages 1404–1409. IEEE, 2002.

[191] I Adji Sulistijono, S Kuswadi, O Setiaji, and N Kubota. Fuzzy control of vision system for humanoid soccer robot efurio. In *Proc.(CD ROM) of the 2011 IFSA (Int. Fuzzy Systems Association) World Congress, Surabaya, Indonesia, pp. FC*, pages 203–1, 2011.

[192] Indra Adji Sulistijono, One Setiaji, Inzar Salfikar, and Naoyuki Kubota. Fuzzy walking and turning tap movement for humanoid soccer robot efurio. In *Fuzzy Systems (FUZZ), 2010 IEEE International Conference on*, pages 1–6. IEEE, 2010.

[193] X Sun, K Hashimoto, A Koizumi, S Hamamoto, T Matsuzawa, T Teramachi, and A Takanishi. Path-time independent trajectory planning of ladder climbing with shortest path length for a four-limbed robot. In *Biomedical Robotics and Biomechatronics (BioRob), 2016 6th IEEE International Conference on*, pages 188–194. IEEE, 2016.

[194] Jeong Sungmoon, Park Yunjung, Mallipeddi Rammohan, Tani Jun, and Lee Minho. Goal-oriented behavior sequence generation based on semantic commands using multiple timescales recurrent neural network with initial state correction. *Neurocomputing*, 129(Complete):67–77, 2014.

[195] G. Syswerda. *A study of reproduction in generational and steady state genetic algorithms*. G. J. E. Rawlings, editor, Foundations of genetic algorithms, Morgan Kaufmann, Indiana University, 1991.

[196] G. Taga, Y. Yamaguchi, and H. Shimizu. Self-organized control of bipedal locomotion by neural oscillators in unpredictable environment. *Biological Cybernetics*, 65:147–159, 1991.

[197] Russ Tedrake, Scott Kuindersma, Robin Deits, and Kanako Miura. A closed-form solution for real-time zmp gait generation and feedback stabilization. In *Humanoid Robots (Humanoids), 2015 IEEE-RAS 15th International Conference on*, pages 936–940. IEEE, 2015.

[198] Gaurav Tevatia and Stefan Schaal. Inverse kinematics for humanoid robots. In *Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on*, volume 1, pages 294–299. IEEE, 2000.

[199] E. Theodorou, E. Todorov, and F. J. Valero-Cuevas. Neuromuscular stochastic optimal control of a tendon driven index finger model. In *American Control Conference (ACC), 2011*, pages 348–355, June 2011.

[200] Robert J. Thompson. Improving round-off in Runge-Kutta computations with gill's method. *Commun. ACM*, 13(12):739–740, December 1970.

[201] Yuichiro Toda and Naoyuki Kubota. Path planning using multi-resolution map for a mobile robot. In *Proc. of SICE Annual Conf.*, pages 1276–1281, 2011.

[202] Yuichiro Toda, Naoyuki Takesue, Kazuyoshi Wada, Naoyuki Kubota, and Hui Yu. Real-time 3D point cloud segmentation using Growing Neural Gas with utility. In *Proc. of Int. Conf. on Human Syst. Interactions*, pages 418–422, 2016.

[203] Duc Trong Tran, Ig Mo Koo, Yoon Haeng Lee, Hyungpil Moon, Sangdeok Park, Ja Choon Koo, and Hyouk Ryeol Choi. Central pattern generator based reflexive control of quadruped walking robots using a recurrent neural network. *Robot. Auton. Syst.*, 62(10):1497–1516, October 2014.

[204] Dong Turk, Damien Kee, Chris Myatt, and Gordon Wyeth. Fuzzy associative memory for humanoid robot joint control. In *Proceedings of Australasian Conference on Robotics and Automation 2005*. Australian Robotics and Automation Association Inc, 2005.

[205] S. Ulbrich, V. R. de Angulo, T. Asfour, C. Torras, and R. Dillmann. Rapid learning of humanoid body schemas with kinematic bezier maps. In *Proc. of Humanoids*, pages 431–438, Dec 2009.

[206] Miomir Vukobratović and Branislav Borovac. Zero-moment point—thirty five years of its life. *International journal of humanoid robotics*, 1(01):157–173, 2004.

[207] Amy Wagoner, Adith Jagadish, Eric T. Matson, Lee EunSeop, Yoanna Nah, Kim Kyeong Tae, Dong Hyung Lee, and Ju-Eun Joeng. Humanoid robots rescuing humans and extinguishing fires for cooperative fire security system using harms' In *Automation, Robotics and Applications (ICARA), 2015 6th International Conference on*, pages 411–415, Feb 2015.

[208] KENNETHJ Waldron and ROBERTB McGhee. The adaptive suspension vehicle. *IEEE Control Systems Magazine*, 6(6):7–12, 1986.

[209] Charles W Wampler. Manipulator inverse kinematic solutions based on vector formulations and damped least-squares methods. *IEEE Transactions on Systems, Man, and Cybernetics*, 16(1):93–101, 1986.

[210] Hongfei Wang, Y.F. Zheng, Youngbum Jun, and P. Oh. DRC-Hubo walking on rough terrains. In *Proc. of the IEEE International Conference on Technologies for Practical Robot Applications (TePRA)*, pages 1–6, April 2014.

[211] Honglun Wang, Wentao Lyu, Peng Yao, Xiao Liang, and Chang Liu. Three-dimensional path planning for unmanned aerial vehicle based on interfered fluid dynamical system. *Chinese Journal of Aeronautics*, 28(1):229–239, 2015.

[212] L-CT Wang and Chih-Cheng Chen. A combined optimization method for solving the inverse kinematics problems of mechanical manipulators. *IEEE Transactions on Robotics and Automation*, 7(4):489–499, 1991.

[213] Stefan Wermter, Cornelius Weber, Mark Elshaw, Christo Panchev, Harry R. Erwin, and Friedemann PulvermÃŒller. Towards multimodal neural robot learning. *Robotics and Autonomous Systems*, pages 171–175, 2004.

[214] Eric R Westervelt, Gabriel Buche, and Jessy W Grizzle. Experimental validation of a framework for the design of controllers that induce stable walking in planar bipeds. *The International Journal of Robotics Research*, 23(6):559–582, 2004.

[215] Darrell Whitley. Genitor: A different genetic algorithm. In *Proc. Rocky Mountain Conference on Artificial Intelligence, Denver, Colorado, 1988*, pages 118–130, 1988.

[216] Daniel E Whitney. Resolved motion rate control of manipulators and human prostheses. *IEEE Transactions on man-machine systems*, 10(2):47–53, 1969.

[217] Martijn Wisse, Arend L Schwab, and Frans CT van der Helm. Passive dynamic walking model with upper body. *Robotica*, 22(6):681–688, 2004.

[218] William A Wolovich and H Elliott. A computational technique for inverse kinematics. In *Decision and Control, 1984. The 23rd IEEE Conference on*, volume 23, pages 1359–1363. IEEE, 1984.

[219] Ching-Chang Wong, Chi-Tai Cheng, Kai-Hsiang Huang, and Yu-Ting Yang. Fuzzy control of humanoid robot for obstacle avoidance. *International Journal of Fuzzy Systems*, 10(1), 2008.

[220] Hataitep Wongsuwarn and Djitt Laowattana. Neuro-fuzzy algorithm for a biped robotic system. *World Academy of Science, Engineering and Technology*, 15:138–144, 2006.

[221] David Wooden, Matthew Malchano, Kevin Blankespoor, Andrew Howardy, Alfred A Rizzi, and Marc Raibert. Autonomous navigation for bigdog. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 4736–4741. IEEE, 2010.

[222] W. Xueli, G. Yapei, and Z. Jianhua. A novel algorithm for shortest path problem based on pulse coupled neural network. In *Proc. of The 27th Chinese Control and Decision Conf.*, pages 2468–2473, May 2015.

[223] S. Yamaguchi and H. Itakura. A modular neural network for control of mobile robots. In *6th International Conference on Neural Information Processing*, volume 2, pages 661–666, 1999.

[224] Simon X Yang and Chaomin Luo. A neural network approach to complete coverage path planning. *IEEE Trans. on Syst., Man, and Cybern., B, Cybern.*, 34(1):718–724, 2004.

[225] Simon X Yang and Max Meng. An efficient neural network approach to dynamic robot motion planning. *Neural Netw.*, 13(2):143 – 148, 2000.

[226] Simon X Yang and Max Meng. Neural network approaches to dynamic collision-free trajectory generation. *IEEE Trans. on Syst., Man, Cybern., B, Cybern.*, 31(3):302–318, 2001.

[227] Simon X Yang and Max QH Meng. Real-time collision-free motion planning of a mobile robot using a neural dynamics-based approach. *IEEE Trans. on Neural Netw.*, 14(6):1541–1552, 2003.

[228] Zhixiao Yang, K. Ito, K. Saijo, K. Hirotsune, A. Gofuku, and F. Matsuno. A combined navigation strategy by a steering wheel and a mouse for a tank rescue robot. In *Robotics and Biomimetics, 2004. ROBIO 2004. IEEE International Conference on*, pages 239–244, Aug 2004.

[229] Zhou Yanjun. Fuzzy omnidirectional walking controller for the humanoid soccer robot. In *IEEE-RAS International Conference on Humanoid Robots*, 2009.

[230] Seung-Joon Yi, Byoung-Tak Zhang, Dennis Hong, and Daniel D Lee. Online learning of a full body push recovery controller for omnidirectional walking. In *Proc. of Humanoids*, pages 1–6, 2011.

[231] Seung-Joon Yi, Byoung-Tak Zhang, Dennis Hong, and Daniel D Lee. Practical bipedal walking control on uneven terrain using surface learning and push recovery. In *Proc. of IROS*, pages 3963–3968, 2011.

[232] Kazuhito Yokoi, Fumio Kanehiro, Kenji Kaneko, Kiyoshi Fujiwara, Shuji Kajita, and Hirohisa Hirukawa. Experimental study of biped locomotion of humanoid robot hrp-1s. In *Experimental Robotics VIII*, pages 75–84. Springer, 2003.

[233] A. Yorita and N. Kubota. Cognitive development in partner robots for information support to elderly people. *IEEE Transactions on Autonomous Mental Development*, 3(1):64–73, 2011.

[234] Y. Yoshida, K. Takeuchi, Y. Miyamoto, D. Sato, and D. Nenchev. Postural balance strategies in response to disturbances in the frontal plane and their implementation with a humanoid robot. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 44(6):692–704, June 2014.

[235] Junying Zhang, Defeng Wang, Meihong Shi, and Joseph Yue Wang. Output-threshold coupled neural network for solving the shortest path problems. *Science in China Series F: Information Sciences*, 47(1):20–33, 2004.

[236] Liyan Zhang, Shuhai Quan, and Kui Xiang. Recurrent neural network optimization for model predictive control. In *(IEEE World Congress on Computational Intelligence). IEEE International Joint Conference on Neural Networks (IJCNN)*, pages 751–757, June 2008.

[237] Yajia Zhang, Jingru Luo, K. Hauser, H.A. Park, M. Paldhe, C.S.G. Lee, R. Ellenberg, B. Killen, P. Oh, Jun Ho Oh, Jungho Lee, and Inhyeok Kim. Motion planning and control of ladder climbing on DRC-Hubo for DARPA robotics challenge. In *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 2086–2086, May 2014.

[238] Yudong Zhang, Lenan Wu, Geng Wei, and Shuihua Wang. A novel algorithm for all pairs shortest path problem based on matrix multiplication and pulse coupled neural network. *Digital Signal Proces.*, 21(4):517–521, 2011.

[239] Jianmin Zhao and Norman I Badler. Inverse kinematics positioning using nonlinear programming for highly artiçulated figures. *ACM Transactions on Graphics (TOG)*, 13(4):313–336, 1994.

[240] 桜井仁 and 内山明彦. インタフェース (2 足歩行ロボット (wabot-1) の開発). バイオメカニズム, 2:208–214, 1973.